NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TOWARDS A LOW-COST QUADROTOR RESEARCH PLATFORM

by

Leon Keith Burkamshaw

March 2010

Thesis Advisor:	Douglas Horner
Second Reader:	James B. Michael

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Towards a Low-Cost Quadrotor Research Platform		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Leon Keith Burkamshaw			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.. IRB Protocol Number_____			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Two aspects of currently available Miniature UAVs (MUAVs) that limit the adoption of this technology for civil and research purposes are the high cost and closed design philosophy. This thesis attempts to solve these problems by presenting an open source design that is focused on low-cost, while maintaining a reasonable level of performance. The use of Commercial Off-The-Shelf (COTS) equipment is maximized where possible to reduce development time and cost. A novel approach used by this design is the use of a Nintendo Wii MotionPlus device as an Inertial Measurement Unit (IMU). This mass produced COTS part provides a three degree of freedom IMU for minimal cost.

All software is of a modular design to ease understanding and facilitate improvements. To reduce development time, and to help discover requirements, a Rapid Application Development (RAD) methodology has been adopted that is suitable for implementation by a single developer. Software prototypes are constructed and iteratively built upon to discover more requirements. At the completion of each phase, testing is performed. Once a suitable level of maturity has been reached, the software prototype is rolled into the main build. Flight-testing is performed at the completion of the design along with a quantitative measure of flight stability.

14. SUBJECT TERMS Rapid Application Development, Quadrotor helicopter, Quadcopter, Inertial Measurement Unit, Digital Signal Processing, Nintendo Wii MotionPlus, dsPIC			**15. NUMBER OF PAGES** 87
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

TOWARDS A LOW-COST QUADROTOR RESEARCH PLATFORM

Leon Keith Burkamshaw
Flight Lieutenant, Royal Australian Air Force
BE (Hons 1st class), University of New South Wales (ADFA), 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2010

Author: Leon Burkamshaw

Approved by: Douglas Horner
Thesis Advisor

James B. Michael
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Two aspects of currently available Miniature UAVs (MUAVs) that limit the adoption of this technology for civil and research purposes are the high cost and closed design philosophy. This thesis attempts to solve these problems by presenting an open source design that is focused on low-cost, while maintaining a reasonable level of performance. The use of Commercial Off-The-Shelf (COTS) equipment is maximized where possible to reduce development time and cost. A novel approach used by this design is the use of a Nintendo Wii MotionPlus device as an Inertial Measurement Unit (IMU). This mass produced COTS part provides a three degree of freedom IMU for minimal cost.

All software is of a modular design to ease understanding and facilitate improvements. To reduce development time, and to help discover requirements, a Rapid Application Development (RAD) methodology has been adopted that is suitable for implementation by a single developer. Software prototypes are constructed and iteratively built upon to discover more requirements. At the completion of each phase, testing is performed. Once a suitable level of maturity has been reached, the software prototype is rolled into the main build. Flight-testing is performed at the completion of the design along with a quantitative measure of flight stability.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1

II. BACKGROUND....3
A. QUADROTOR HELICOPTERS....3
1. Advantages of the Quadrotor Helicopter....4
2. Quadrotor Applications....4
B. EXISTING QUADROTOR DESIGNS....5
C. OUTLINE OF THE PROPOSED SOLUTION....6

III. RISK MANAGEMENT....9
A. INTRODUCTION....9
B. IDENTIFY....9
1. Quadrotor Frame....9
2. Inertial Measurement Unit Data Processing....9
3. Requirements Change....10
4. Low-Cost COTS Inertial Sensors....10
5. Available Time....10
C. ANALYZE....10
1. Quadrotor Frame....11
2. Inertial Measurement Unit Data Processing....11
3. Requirements Change....11
4. Low-Cost COTS Inertial Sensors....12
5. Available Time....12
D. PLAN....12
1. Quadrotor Frame....12
2. Inertial Measurement Unit Data Processing....13
3. Requirements Change....13
4. Low-Cost COTS Inertial Sensors....14
5. Available Time....14
E. TRACK....15
F. RESOLVE....15

IV. HARDWARE DESIGN....17
A. INTRODUCTION....17
B. ELECTRONICS SYSTEM DESIGN....17
1. Inertial Sensor(s)....18
2. Battery....18
3. Radio Control (RC) Receiver....19
4. Motor....19
5. Electronic Speed Controller (ESC)....20
6. On-Board Processor....20
7. External Flash Memory....21
C. FRAME DESIGN....21
D. MATERIAL COST....22

V. SOFTWARE METHODOLOGY 25
A. INTRODUCTION 25
1. Assign Software Modules from High Level Requirements 27
2. Build Prototype for a Single Software Module 27
3. Implement Any Newly Discovered Requirements for the Module 27
4. Test Software Module 28
5. Repeats Steps 2–4 Until No More Failures Are Found 28
6. Integrate Module Into Core Software Build 28
7. Perform Integration Testing 28
B. SOFTWARE MODULES 29
C. FAULT DETECTION 31
VI. SOFTWARE DESIGN 33
A. INTRODUCTION 33
B. SOFTWARE FLOW 33
1. Interrupts 33
2. Flow Diagrams 35
C. DIGITAL SIGNAL PROCESSING 37
1. Augmented Stability Control Loops 37
2. Inertial Measurement Unit (IMU) Data Processing 38
3. Finite Infinite Response (FIR) Filters 42
D. SOFTWARE TESTING 47
1. Test 1 48
2. Test 2 49
3. Test 3 49
4. Test 4 50
E. PC BASED CONFIGURATION SOFTWARE 51
1. Fast Adjustment of Internal PID Values 52
2. Initiate and Confirm RC Transmitter Calibration 53
3. Initiate and Confirm Gyro Calibration 53
4. Calibrate ESCs 53
5. Manage External Flash Memory 53
VII. TUNING AND TETHERED TESTING 55
A. INTRODUCTION 55
B. TEST RIG 55
C. TETHERED TESTING 56
VIII. FLIGHT TEST 59
A. INTRODUCTION 59
IX. CONCLUSION 63
LIST OF REFERENCES 65
INITIAL DISTRIBUTION LIST 67

LIST OF FIGURES

Figure 1. Quadrotor schematic showing rotor direction of rotation (From [2]) 3
Figure 2. Toy quadrotor: Walkera UFO (from Walkera [Online]) 5
Figure 3. Electronic system block diagram 17
Figure 4. Quadrotor cost breakdown pie chart 23
Figure 5. UML Diagram of quadrotor software 30
Figure 6. Software flow for main function 35
Figure 7. Software flow for mainLoop module 36
Figure 8. Rotation rate limiter for a single axis 37
Figure 9. Heading Hold algorithm (From [8]) 38
Figure 10. Pitch gyro bias drift 40
Figure 11. Pitch gyro angular drift 40
Figure 12. Yaw gyro bias drift 41
Figure 13. Yaw gyro angular drift 42
Figure 14. FIR filter basic structure (From [15]) 43
Figure 15. Pitch gyro spectrum with induced mechanical vibration 44
Figure 16. dsPIC Lite filter program 45
Figure 17. Roll gyro samples showing mechanical noise 46
Figure 18. Filtered roll gyro samples 46
Figure 19. LabVIEW test program screen capture 48
Figure 20. PC configuration GUI 52
Figure 21. Quadrotor test rig 55
Figure 22. User input for first flight 59
Figure 23. Quadrotor in flight 61

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Risk Matrix (From [6]) 11
Table 2. Quadrotor cost breakdown 22

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis presents a design for a low-cost quadrotor research platform. Chapter I gives an introduction to existing UAVs and their applications. In addition, it discusses limitations of currently available UAVs and an outline of the proposed solution.

Chapter II gives a background on quadrotor helicopters and their applications across varying industries. Existing quadrotor designs are presented and their individual shortcomings are highlighted. Finally, this chapter presents the proposed solution that is the remainder of this thesis.

The implementation of a risk analysis and management system is discussed in Chapter III. This chapter identifies five major risks for this thesis and discusses these risks at each stage of the risk management process.

The hardware design is covered in Chapter IV. This chapter breaks the hardware design down into both electronics system design and frame design. The electronics system design section discusses the selection of electronic parts and modules while the section on frame design presents the available options for a suitable quadrotor frame.

Chapter V presents the Rapid Application Development (RAD) software methodology adopted for this thesis. The implementation details of a RAD methodology, suitable for implementation by a single developer, are presented in detail along with a list of software modules.

The software design and encountered challenges are presented in Chapter VI. Flow diagrams are used to provide a functional flow of the software. The Digital Signal Processing performed by the on-board processor is discussed along with the control algorithms implemented to provide augmented stability. Software testing is a large part of the development effort and is also presented in this chapter.

The control algorithms implemented for this design require tuning before flying. The selected tuning procedure is outlined in Chapter VII along with a description of the tethered test flights tuning verification.

The first flight test is discussed in Chapter VIII and logged flight data is presented in graphical form to give a quantitative measure of stability. Performance predictions are calculated, based on flight data, and comparisons are made to existing designs.

Chapter IX concludes the thesis with an overview of what has been achieved and some suggested improvements. Ideas for future development are also presented in this chapter.

LIST OF ACRONYMS AND ABBREVIATIONS

ADC	Analog to Digital Converter
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DC	Direct Current
DMA	Direct Memory Access
DOF	Degrees Of Freedom
DSP	Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
ESC	Electronic Speed Controller
FHSS	Frequency Hopping Spread Spectrum
FFT	Fast Fourier Transform
FIR	Finite Infinite Response
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
ISM	Industrial Scientific Medical
I2C	Inter-Integrated Circuit
LiPo	Lithium Polymer
MEMS	Microelectromechanical System
MMC	Multimedia Memory Card
MUAV	Miniature Unmanned Aerial Vehicle
PCM	Pulse Code Modulation
PD	Proportional Derivative
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
PPM	Pulse Position Modulation

PVC	Polyvinyl Chloride
RAD	Rapid Application Development
RC	Radio Controlled
RTOS	Real Time Operating System
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing

ACKNOWLEDGMENTS

I would like to thank the Royal Australian Air Force (RAAF) for providing me with the invaluable opportunity to undertake a Masters level program in an overseas posting. I have no doubt that the knowledge I have gained throughout this program will be of great benefit to the RAAF.

I would also like to thank my thesis supervisor, Research Associate Professor Doug Horner, for the chance to work on such an exciting project and for his support.

My greatest gratitude goes to my family, my wife, Liz and son Will, for always bringing a smile to my day.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Over the past 10 years, the UAV market has grown rapidly and it is expected that this market expansion will continue for the foreseeable future. While much of this growth is attributed to defense applications, there are an increasing number of applications for UAVs in the commercial sector. This is particularly so for smaller sized UAVs categorized as Miniature UAVs. Miniature UAVs or MUAVs range in size from Micro Air Vehicles (MAVs) to a "man portable" size [1].

While MUAVs are gaining popularity, there are some aspects of current designs that are limiting their adoption. The first problem is cost. Current MUAV designs are built from scratch and use customized airframes and electronics, which dramatically increases cost. Another aspect of current designs that limits MUAV adoption, especially in the field of academic research, is the lack of access. When using MUAVs for research it is advantageous to have low-level access to both autopilot source code and hardware schematics. This makes it easier to integrate new sensors, using existing processing power, as well as experimenting with new algorithms.

This thesis attempts to address these problems of cost and lack of access through the design and development of a quadrotor MUAV for possible defense, commercial and research applications. At the same time, it attempts to demonstrate how a Rapid Application Development method can be used in practice by a sole developer. The design goal is to produce a capable low cost quadrotor helicopter by utilizing mass-produced Commercial Off-The-Shelf (COTS) hardware. While full flight autonomy is the objective of this design, the first stage in achieving this is to implement an augmented flight control system that provides enough stability so the quadrotor can be remotely piloted.

Although the design of a quadrotor helicopter encompasses many disciplines including mechanical, aeronautical, electronic, and software engineering this design will use COTS parts for the mechanical and electric hardware where possible and concentrate on software engineering aspects. This thesis takes a novel approach to the use of a mass produced COTS part made by Nintendo called the Wii MotionPlus. While this device is

designed to plug into a Nintendo Wii controller, and provide a more real game experience, its benefit in this thesis is the 3-axis rotation rate information it provides at a cost of only $20.

In order to identify and track significant risks for the project and provide trigger points for identifying when a risk becomes an issue an initial risk assessment is undertaken of the proposed design. This provides a mechanism to solve problems before they cause significant delays or incur any additional cost.

Once the hardware design is complete, the software engineering aspects of the design need to be defined. The quadrotor helicopter inherits all of its functionality through software and hence is a relatively software intensive system. Given the amount of software required for such a design, coupled with limited development time, a Rapid Application Develop (RAD) methodology is used. The RAD approach allows the software coding to commence as early as possible with the limited number of requirements that are initially available. The approach taken is a method that could be sensibly adopted by a single developer.

Any software implementation creates challenges and problems that need to be addressed. Some of these problems discussed include the selection of an appropriate filtering and control algorithm. Software testing is an important part of the development and the testing methods employed within the development are also outlined.

Before the flight control system can provide augmented stability, it requires tuning. This tuning method is discussed in detail along with a tethered flight test to validate the tuning. Lastly, the flight test results are used to provide a quantifiable measurement of stability and performance predictions are made from the given test data with comparisons made to MUAV commercial solutions.

II. BACKGROUND

A. QUADROTOR HELICOPTERS

A quadrotor helicopter consists of four rotors that are mounted at the end of two perpendicular axes. The fixed pitch rotors are driven by a DC electric motor, either directly or through a reduction gearbox. Rotors at opposite ends of an arm turn in the same direction while rotors on a perpendicular axis rotate in the opposite direction. This concept is illustrated in Figure 1.

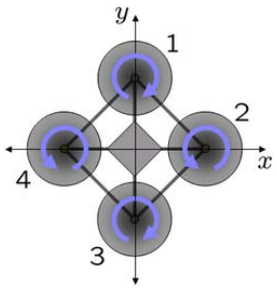

Figure 1. Quadrotor schematic showing rotor direction of rotation (From [2])

When all four motors are spinning at the same speed, the rotors create thrust that lifts the quadrotor into the air. As there are pairs of rotors spinning in opposite directions, the torque produced in each direction around the yaw axis cancels out and the yaw angle remains constant. To change the pitch attitude, the speed of motor 1 is reduced while the speed of motor 3 is increased, or vice versa, creating a non-zero pitch angle. As both motor 1 and motor 3 are rotating in the same direction the total counteracting torque provided is not changed so the quadrotor maintains its yaw angle. The roll attitude is adjusted in a similar manner. To adjust the yaw angle the speed of motors 1 and 3 are increased while the speed of motors 2 and 4 are decreased, or vice versa. This creates an imbalance in the total torque in the yaw axis and so the quadrotor will change yaw angle. The quadrotor should maintain a relatively constant thrust during yaw and the height of the aircraft should remain constant.

While the above description provides a simplified overview of how a quadrotor maneuvers, the dynamics of the aircraft are complex and tightly coupled [3], [4]. These dynamics make it extremely difficult for a human to control the quadrotor without an on-board flight augmentation system to reduce the aircraft response down to an acceptable level.

1. Advantages of the Quadrotor Helicopter

Despite the complex control systems required for a quadrotor aircraft, there are many benefits to this aircraft over other platforms. Having four rotors as opposed to a single rotor in a traditional helicopter allows each of the rotors on a quadrotor to be smaller and lighter, hence carrying less kinetic energy. This is advantageous when working within indoor environments. For example, in the undesired case of a blade striking an object, due to the design of a quadrotor, much less damage will result when compared to a helicopter in the same situation. It is also possible to mount the rotors within a duct or shroud to protect both the aircraft and any object if contact occurs.

Due to the complex electronic control system, the mechanics of a quadrotor are relatively simple and the aircraft is able to use fixed pitch propellers. This reduces setup, maintenance, and manufacturing costs and time associated with a quadrotor. The relatively simple mechanical setup of a quadrotor also leads to limited vibration making it a friendly environment for inertial sensors and cameras.

2. Quadrotor Applications

Quadrotor aircraft are suitable for a wide range of applications such as aerial photography, law enforcement, Defense, academic research, and as a teaching aid for control theory.

MUAVs are becoming more popular for both use in commercial industry and Defense. MUAVs are now carried by small teams of defense personnel and these aircraft offer a remote look at what lies beyond the horizon. One of the current issues with MUAVs used for this purpose is launch and recovery. This is because only little ground space is often available combined with the possibility of being surrounded by trees. The quadrotor is ideal

in this type of environment as it is a Vertical Take Off and Landing (VTOL) aircraft and is easy to fly due to the onboard flight augmentation.

The quadrotor helicopter can also make an excellent teaching aid for control theory as it provides a real world control problem. Students can make changes in the control algorithms and feel the effect of those changes when flying. The quadrotor can also be used in the teaching of dynamics. The dynamic calculations and simulations can be performed and then qualified against real flight data from the inertial sensors.

B. EXISTING QUADROTOR DESIGNS

While the quadrotor's popularity has only increased in recent years, there are many existing designs available. These designs can be broken down into two main categories: toys and professional commercial products.

There are numerous quadrotor designs available as a Radio Controlled (RC) toy. Some examples are the Walkera UFO #5, Walkera UFO #8, Dragonfly, and Alien Air Jump Jet.

Figure 2. Toy quadrotor: Walkera UFO (from Walkera [Online])

While these toys are very low cost and provide a full six Degrees Of Freedom (DOF) Inertial Measurement Unit (IMU) they lack robustness and can only carry very light payloads. Personal experiences with the quadrotor in Figure 2 lead to conclusions that the inertial processing was also not optimal. This sub-optimal behavior was

experienced after approximately one minute of flying where full deflection on a control axis is sometimes required to maintain a zero degree pitch and roll. Given the six DOF sensor of this quadrotor, the expected behavior is to auto-level to zero degrees pitch and roll when no control input is applied. In order to keep costs down, the toy versions also use brushed DC motors as opposed to brushless motors. This leads to significantly lower power efficiency. Brushless motors, while more expensive, also have a much longer service life as they do not have brushes that undergo mechanical wear.

Quadrotor helicopters are also available as a ready to fly commercial product for professional use. One such aircraft is the Hummingbird quadrotor from Ascending Technologies. While the hummingbird performs very well, it is expensive and, importantly for research use, there is no source code or hardware schematics given for the low-level autopilot. While the hummingbird does provide a high level API for accessing some functions there is no way to modify the Inertial Measurement Unit (IMU) processing algorithms, control loop algorithms, and controller input processing from the Radio Controlled transmitter. This places limitations on the availability of this aircraft to research institutions as well as the scope of research that can be performed.

C. OUTLINE OF THE PROPOSED SOLUTION

The improved quadrotor solution outlined by this thesis will maximize the use of mass produced COTS equipment to reduce cost while also making available both the hardware schematics and software for researchers to modify. Ideally, the cost of this aircraft should be as low as possible without sacrificing performance.

Most of the work for the quadrotor will be focused on software to allow control of the quadrotor. Due to the limited time available, along with the lack of complete requirements, a Rapid Application Development (RAD) approach to software development will be implemented. This methodology uses software prototypes as a starting point and grows the maturity of the prototype as the development progresses. Through this process, more requirements are discovered and incorporated into the prototype. Software testing will also play a critical role in this development, as there is kinetic energy in this system that can cause harm to the system itself as well as people.

Developed as a research platform, it is highly likely that the software will be modified and built upon in the future. For this reason, the software needs to have a high level of maintainability.

THIS PAGE INTENTIONALLY LEFT BLANK

III. RISK MANAGEMENT

A. INTRODUCTION

The multitude of in-depth skills required to build a quadrotor coupled with limited time presents a unique challenge for thesis research. Taking on such a project means that a risk analysis and system management is critical. Risk was managed in this thesis using the five essential elements of the risk management process [5].

- Identify
- Analyze
- Plan
- Track
- Resolve

Using the five essential elements outlined above the risk management process was initiated from the start of the project. Each step of the process is discussed below.

B. IDENTIFY

The first step in the process is to identify areas of risk within the project. These areas were identified based on confidence level of achieving the stated goal.

1. Quadrotor Frame

The quadrotor frame must be light, rigid, low-cost, and crash resistant. The ability to design and create the quadrotor frame was considered a risk due to lack of experience in building quadrotor frames and limited mechanical skills.

2. Inertial Measurement Unit Data Processing

The quadrotor will require some inertial sensing to determine its attitude and hence allow the Digital Signal Processor (DSP) to compensate for input that is not given by the human controller. To extract attitude information from the inertial sensors usually requires Kalman filtering, an area which can be complex to understand and implement.

3. Requirements Change

While the basic requirements for this design were established, it is likely that many smaller requirements are not initially identified. Additionally, it is highly likely that existing requirements will change as the development progresses.

4. Low-Cost COTS Inertial Sensors

As one of the main requirements for this design is low cost, a mass-produced COTS inertial sensor must be used. These devices are prone to drift dramatically with temperature and it is unknown if they are suitable for this application.

5. Available Time

This is possibly one of the greatest risks in the project. It is uncertain if there will be enough time to complete the entire design and still have enough time to produce a quality thesis document. It is hard to estimate exactly how long the entire development will take (there is generally a tendency to underestimate) as well as how much time will be more generally available.

C. ANALYZE

In this second stage, the identified risks are analyzed to obtain an estimation of the probability of the risk becoming an issue and the severity of impact if the risk becomes an issue.

Once the probability of occurrence and severity are obtained the risk matrix in Table 1 is used to identify the resultant risk category.

Probability Of Occurrence	Insignificant	Minor	Moderate	Major	Severe
Highly Likely	M	H	H	E	E
Likely	M	M	H	H	E
Possible	L	M	M	H	E
Unlikely	L	M	M	M	H
Rare	L	L	M	M	H

Table 1. Risk Matrix (From [6])

1. Quadrotor Frame

Given the mechanical ability and experience required in such an undertaking, the probability of this risk becoming an issue was determined to be "Likely." The severity of this occurring is Major, as without a frame the quadrotor will not fly. Even if the frame was built but it had some defect (i.e., was not perfectly symmetrical) this could have a major impact on the controllability. The resultant category for this risk is "High."

2. Inertial Measurement Unit Data Processing

While lack of experience in IMU data processing was a concern, an engineering background provides some confidence that this could be learned and implemented. For this reason, the probability of this risk becoming an issue was determined to be Possible. In the worst case that no IMU processing algorithm could be implemented, the severity is Major as the on-board controller requires this information to provide augmented stability. The resultant category for this risk is "High."

3. Requirements Change

It is extremely difficult to have a concrete final set of requirements for a product not previously built. Additionally it is inevitable that requirements will change. This is especially the case in the areas of research and development. The probability of this risk

becoming an issue was defined as almost certain. While the consequence of requirements change is not likely to halt the development, it can lead to time delays and additional costs. The consequence of changing requirements is considered to be moderate. The resultant risk category is “High.”

4. Low-Cost COTS Inertial Sensors

MEMS based inertial sensors have previously been used in quadrotor control systems but the selected sensor for this design is untested in the mechanically noisy environment of a quadrotor helicopter. For this reason, the probability that this risk becomes an issue was determined to be Possible.

If the inertial sensor were deemed unsuitable, another sensor would have to be selected. This would result in additional cost and a short delay for transport of the item but the overall impact on the project is minimal. The severity of impact was determined to be “Minor.” The resultant risk category is “Medium.”

5. Available Time

The probability of occurrence for this risk was determined to be “Possible.” While it is highly desirable to complete the initial proposal, if time can be justified and the work makes a positive contribution, then the thesis can still be termed successful. For this reason, the impact was determined to be Moderate. The resultant risk category is Medium.

D. PLAN

The planning stage identifies alternatives for unacceptable risk category items and provides an action plan for the remaining risks. For this thesis project, any risk with a category of High or greater was considered unacceptable and some form of risk removal or mitigation needed to be implemented.

1. Quadrotor Frame

The resultant risk category of High for this item was unacceptable. Ideally, this risk would be removed altogether by purchasing a COTS frame but there is no suitable

low cost solution available. The next best option was to copy an existing design that was made of low cost materials and easily constructed. A simple and elegant design was found and selected for the quadrotor frame. This approach aligns with modern business practice of only performing core business processes. In this case, the core business is software development using a rapid application methodology not mechanical design and construction.

The revised probability of occurrence for this risk was then considered unlikely. This changes the revised resultant risk category to medium. This was considered an acceptable level of risk. The action plan if this risk still became an issue was simply to buy a more costly COTS solution and work around the unsuitable characteristics of the frame.

2. Inertial Measurement Unit Data Processing

The resultant risk category for this risk was unacceptable. The underlying reason for this risk was the lack of understanding of IMUs and IMU data processing. This triggered considerable research in this area. The conclusions made from the research were that while Kalman filtering and a 6-DOF inertial sensor are preferred and often provide significant stability, there are much simpler algorithms that provide suitable performance. Additionally only a 3-axis rate gyro is required for a bare minimum augmented flight control system.

To reduce this risk, the decision was to use digital filtering of only a 3-axis rate gyro sensor for the initial design stage. Time permitting a 3-axis accelerometer would be added with more complex filtering algorithms, such as the Kalman filter. The revised probability of occurrence for this risk was determined to be Unlikely. The revised resultant risk category of medium was accepted.

3. Requirements Change

The resultant category for this risk was unacceptable. Requirements changes are inevitable in most projects and this is even more likely for academic research work.

Aside from applying due diligence to the early design and specification stage, in this project there was little that can be done to counteract the possibility of changes to the initial specification.

Another approach is to also constantly expect and preempt changing requirements as the project progresses and plan for each possible outcome. Where a modular design is used, then the project will also be more adaptable to changing requirements. It was not possible to change the probability of occurrence of this risk but the impact could be minimized by some of the previously mentioned methods. On revision of this risk, it was decided to leave the severity at Medium and accept the risk and plan for constant requirements change.

4. Low-Cost COTS Inertial Sensors

The resultant category for this risk was considered acceptable but an action plan was required to provide a resolution if the risk were to become an issue. The action plan for this risk was to purchase the proven sensors in the event that the chosen sensors were not suitable for the quadrotor application. In this instance, the additional cost would have to be accepted. This risk was accepted at the current level with the specified action plan.

5. Available Time

The resultant risk category for this risk was considered acceptable but constant management of the project would be required to track progress and provide an early warning to a sliding schedule.

Prioritizing features for incorporation into the quadrotor plays an important role in managing the available time. The initial design will be as simple as possible and, if time permits, more complex features will be added, such as full automation. By taking the time to produce a quality schedule for the entire project, including several significant milestones, this provides a reference point to track project progress.

E. TRACK

Once the risk planning is completed, the risks need to be tracked as the project progresses. Once a week all risks are reviewed to identify any apparent issues or risk escalations. In addition, the time schedule was also reviewed to obtain an indication of project progress. This was used as a trigger for risk five. If new requirements were introduced or identified, an investigation into impact on all aspects of the project was carried out and an estimate of the required additional work was performed.

F. RESOLVE

This final process in the risk management model is used to resolve risks that become an issue. If the tracking stage identified an impending issue, the alternative plan that was identified at the planning stage, is implemented. Once the alternate plan is carried out, the risk is again analyzed through the other four processes in the risk management model for reevaluation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. HARDWARE DESIGN

A. INTRODUCTION

The hardware design is broken down into two subcategories:

- Electronics system design
- Frame design

Each of these designs is discussed below.

B. ELECTRONICS SYSTEM DESIGN

To enable a human pilot to control the aircraft, the quadrotor helicopter requires some form of on-board control system to provide augmented stability. A block diagram of the electronics system is contained in Figure 3.

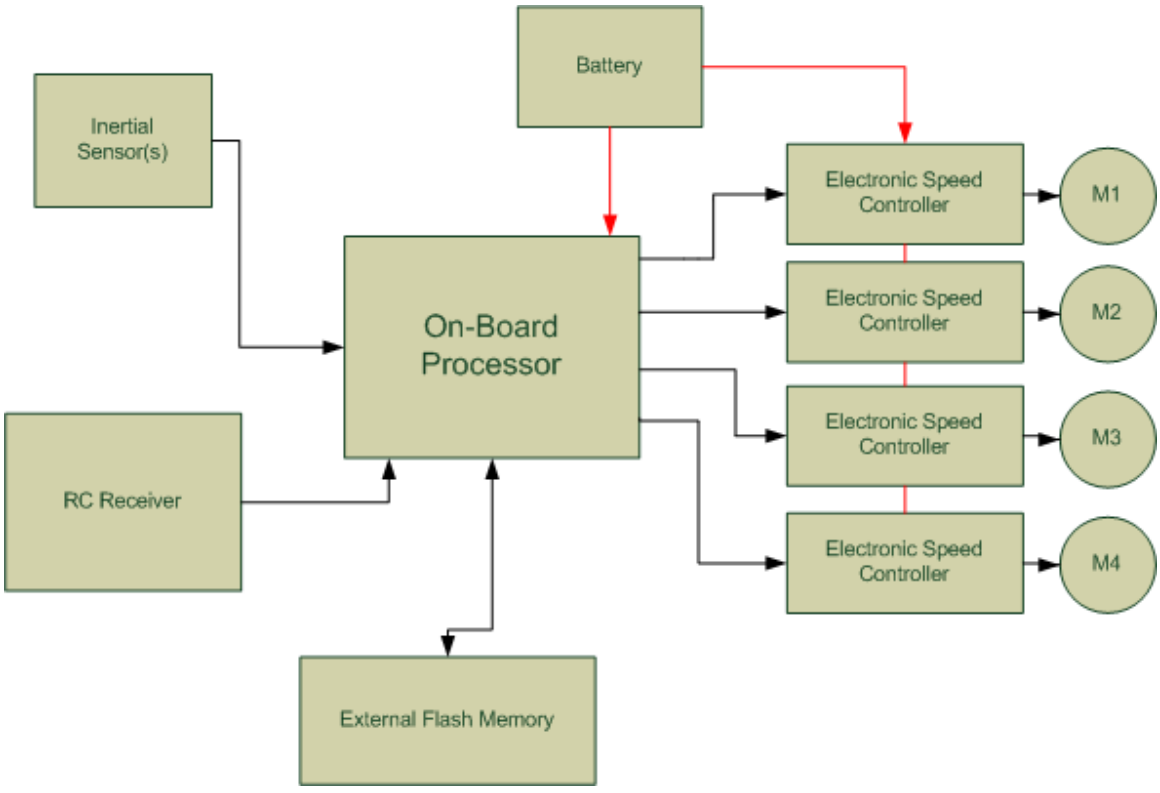

Figure 3. Electronic system block diagram

A primary design constraint for this system was that it would be low cost. To achieve this goal the use of COTS equipment was maximized where possible.

1. Inertial Sensor(s)

The inertial sensors are often the single most expensive electronic component in a quadrotor design. Older quadrotor designs such as the X-UFO made by Silverlit Electronics used small mechanical gyros. These were relatively cheap due to low-cost labor, but suffered from mechanical failure and were not crash tolerant.

Modern quadrotor designs use Microelectromechanical (MEMS) based inertial sensors. These sensors consist of very small mechanical components ranging from 1 to 100 micrometers in size and fit inside a traditional electronic chip package [6]. The cost of these sensors in low quantities has made these sensors relatively expensive compared to the other electronic components required for a quadrotor. Some new generations of MEMS sensors are low in cost for small quantities but are not suitable for quadrotor platforms due to their low resonant frequency. The vibration frequency of the quadrotor is in the same range as the low-cost MEMS resonant frequency, this produces large errors from the sensors that are unable to be filtered.

To reduce the cost of the inertial sensor, this project uses a mass produced COTS product called a Wii MotionPlus, made by Nintendo. This device contains a single chip dual axis MEMS gyroscope that provides pitch and roll information, as well as a single axis MEMS gyroscope to provide yaw information. Additionally, the device contains analog conditioning circuits for the gyros and a microcontroller with built in ADC. The device is interfaced to the main control board through an Inter-Integrated Circuit (I2C) serial bus. All of this is accomplished for the total cost of approximately USD$20.

2. Battery

The battery selected for the design is a Lithium-Polymer (LiPo) battery. While NiCad and NiMh chemistry batteries are available at lower cost, the LiPo has a much

higher energy density and their cost is constantly falling. A 3S LiPo was selected (meaning 3 LiPo cells in series) as the selected motor and propeller combination works efficiently at this voltage.

3. Radio Control (RC) Receiver

A wide range of RC receivers is available with various protocols and frequencies. It should be noted that receivers are not all created equally. Even receivers of the same frequency with the same protocol can have very different performance specifications.

Until recently, RC receivers operated in the VHF band and used either Pulse Position Modulation (PPM) or Pulse Code Modulation (PCM). PCM was more robust as it sent error checking information with each packet but it also suffered from "lockout." Lockout occurred when synchronization was lost with the receiver, for whatever reason, and the radio-controlled model would not respond to the controller. This is obviously a highly undesirable state. Modern RC control equipment uses the 2.4GHz Industrial Scientific Medical (ISM) band and uses a more robust modulation method than PPM or PCM.

Newer systems use either Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS). One of the greatest advantages of the 2.4GHz spectrum used by modern RC systems is that the electrical noises from sources such as electric motors are not present. A 2.4GHz system was chosen (made by Spektrum) for the quadrotor design because of the lower noise attributes of the 2.4GHz band mentioned above and the robust communications provided by DSSS.

4. Motor

There are two basic types of technologies to choose from, Brushed DC and Brushless DC motors. Brushed DC motors are very low cost, have a high starting torque, but suffer from poor efficiency due to the use of mechanical commutation. Brushless DC motors are higher in cost, require an electronic controller, have a low starting torque, but are much more efficient due to electrical commutation.

A brushless DC motor was chosen for this design due to the high efficiency. The brushless motor selected was an “outrunner” design that is optimized for running direct drive to a propeller, negating the need for a gearbox. This saves weight, increases drive efficiency, reduces maintenance and reduces cost.

5. Electronic Speed Controller (ESC)

As brushless motors are being used in this design, the ESC required is much more complex than that required for a brushed motor design. The Hummingbird quadrotor, made by Ascending Technologies, uses a specially designed ESC that enables fast updates of motor commanded speed (1kHz). While this enables the control loop to run faster there is additional cost in producing a custom ESC for this purpose.

Standard brushless RC ESCs were chosen for this design in an attempt to keep costs to a minimum. The disadvantage of using ESCs designed for regular RC model use is the update rate for motor speed selection is limited and the exact rate is unknown.

6. On-Board Processor

The on-board processor used by existing designs varies from simple 8-bit microcontrollers to 32-bit processors and FPGAs. For the initial control software, only simple control algorithms are planned that will require minimal processing ability. Despite this, the design of this quadrotor is pitched at research institutions and so requires considerable capacity for more complex processing while still mindful of cost constraints.

A Microchip dsPIC development board [7] was chosen for the design as it meets both requirements of being low cost while still providing ample processing power. This processor has many on chip communication peripherals such as Serial Peripheral Interface (SPI), I2C, and Universal Synchronous Asynchronous Receive Transmit (USART) ports that facilitates raid incorporation of new sensors.

An important consideration in the selection of processor is the availability of compilers and Integrated Development Environments (IDEs). Microchip provides an academic version of their compiler for the dsPIC line of processors at no cost.

An even more important consideration when choosing microcontroller devices and development environment is previous developer experience with the particular device and environment. This has the ability to dramatically reduce development time, particularly in the early stages, as the developer does not have to learn the particulars of the device and IDE. Having previously worked with the Microchip dsPIC line of digital signal controllers as well as having experience with Microchip's MPLAB development environment made the dsPIC an obvious personal choice.

7. External Flash Memory

The external flash memory is available to store configuration parameters used by the controller as well as for storage of real time logged data. The ability to store configuration parameters enables changes to configuration parameters to be made without the requirement of recompiling the firmware and re-flashing the dsPIC microcontroller.

The data logging capability will be useful for both initial development as well as for future research. The current flash memory is a 2MB 8pin DIP package Integrated Circuit (IC) that uses a Serial Peripheral Interface (SPI) port for communicating with the dsPIC. This device supports high-speed burst writing of data and typically 1Mbps overall. Once the driver software is written for this device larger capacity SPI flash memory could also be used with the potential for high capacity SD/MMC cards.

C. FRAME DESIGN

The primary design requirements for the frame are that it is light as possible as well as being rigid. The lighter the aircraft, the less power that will be required to lift it. This translates into longer run times and a greater power margin for additional payloads. The frame needs to be rigid enough so as to minimize the amount of flex during flight, as this is an assumption made by the control algorithms. The control system will tolerate flex in the frame but at the expense of the effectiveness of the control system.

Existing frames are commercially available that meet the above requirements well but these cost several hundred dollars. To keep construction costs down, an open hardware design was chosen that can be constructed for approximately $40–$60 and only

requires some simple power tools [7]. This open design was modified to provide a lighter mounting plate for the dsPIC board as well as providing greater vibration dampening. In addition to providing cost-effective construction, this design also allows for simple low cost repairs if damage occurs.

D. MATERIAL COST

The cost breakdown for the quadrotor can be seen in Table 2.

Item	**Cost**
Frame – Trex 600 Tail Boom	$20
Frame – PVC Pipe 4-Way joint	$3
Frame – Fasteners, Mounts, Miscellaneous	$20
Frame – Trex 600 Landing gear	$20
dsPICDEM Microcontroller Dev Kit	$80
Wii MotionPlus	$20
4 x Brushless motors	$24
4 x Brushless Electronic Speed Controller	$47.60
Rhino 2150mAh LiPo Battery	$20
4 x Counter Rotating Propellers (10x4.5)	$10
Spektrum AR6100 2.4GHz receiver	$49
Total	**$313.60**

Table 2. Quadrotor cost breakdown

At a total cost of $313.60, including the receiver, this is well within the low-cost design goal when compared to commercial offerings. To provide full autonomous operation the addition of a GPS, 3-axis accelerometer, and magnetometer would also be required. Purchasing these items as pre-soldered modules would only cost an additional $150.

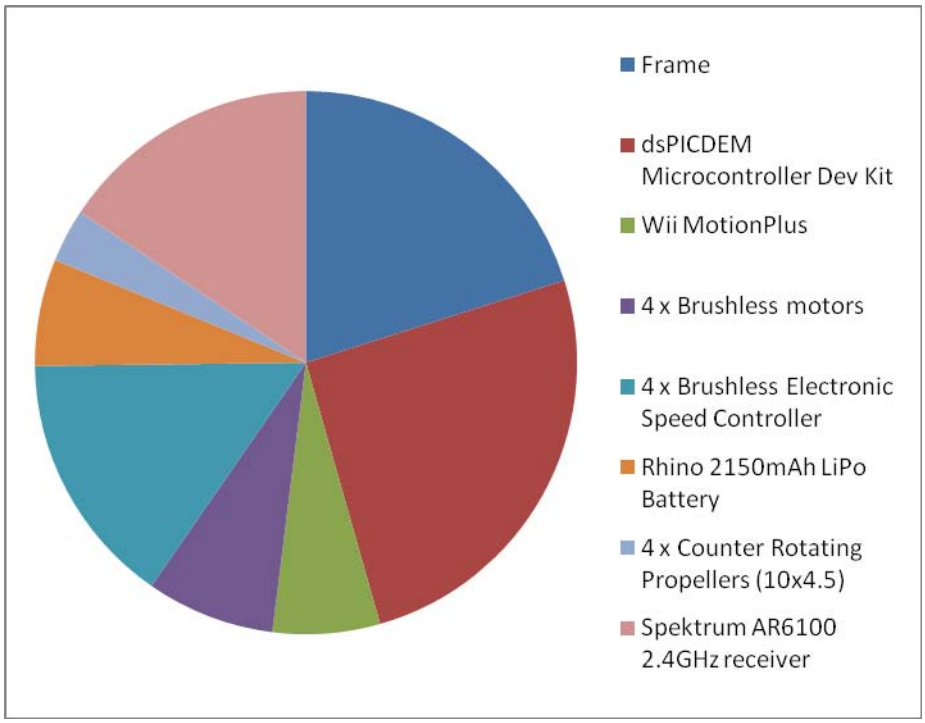

Figure 4. Quadrotor cost breakdown pie chart

Figure 4 gives a graphical representation of cost breakdown. From the pie chart, the dsPICDEM microcontroller development kit is the largest contributor to cost, followed by the frame. While a lower cost microcontroller board could have been used, the selected board provides large scope for future development. Despite the frame being of simple construction, the relative cost compared to other parts for the quadrotor is significant. Time permitting a more refined hardware design will be performed that further reduces the cost of the frame.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SOFTWARE METHODOLOGY

A. INTRODUCTION

A quadrotor helicopter requires experience in many disciplines. Of these areas, the greatest proportion of effort was in developing software. This was exacerbated by the fact that a full set of requirements for the quadrotor was not available at the beginning of the project, due to lack of experience in building such a product. Given this lack of full requirements a traditional software development methodology, such as the Waterfall model could not be used. Due to the time constraint and lack of requirements, a Rapid Application Development (RAD) approach was adopted [10].

The RAD approach uses both iterative development and prototyping to gradually produce a final product. To allow the software to have a high level of maintainability the software is of a modular design. Adopting a modular design also enables the rapid incorporation of additional sensors to the processor board by simply adding another file to the build that abstracts the new hardware.

There are many existing forms of RAD including Extreme Programming (XP) and Scrum [10]. While these use some form of iterative development they are intended for development teams of at least two people, and in some cases, many more. XP has indeed been used successfully as the software methodology for university projects consisting of two or more people [11]. What is required, however, is a form of RAD that assists in the discovery of requirements, allows for requirements change, and can be followed by a single developer. Individual developers with no internal reporting requirements may not feel the need for the adoption of a formal methodology. However, as a contractor to government or business or even as a matter of basic professionalism, having in place some form of methodology allows for development to occur without the need for extensive pre-planning but still using a disciplined, systematic and quantifiable approach. For smaller projects that are required within short time frames, government or business may contract directly with individual developers, which allows for direct communication and collaboration with the developer.

However, as the field of software engineering continues to grow in importance, it may become difficult for even an individual developer to be successful in contracting for tenders without being able to demonstrate the methodology to be used in developing the product. And for those contracts not requiring any formal tender documents (which are often the smaller projects undertaken by individual developers), it is likely to give clients a level of confidence where developers can demonstrate a systematic approach to development.

The RAD methodology is clearly suitable for smaller projects tackled by individual developers where timeframes are limited and clients have not had the time to consider or unaware of specific requirements. Software can be written quickly allowing for changing requirements.

The proposed RAD attempts to provide an approach specifically for a single developer. While any methodology could be discussed at length in the abstract, there is no better way to demonstrate and test the usefulness and practicality of such a methodology than through application. The development of the quadrotor in this thesis by a single developer demonstrates how the proposed new RAD methodology can be applied.

The proposed new methodology will use the following steps to produce the final software product:

1. Assign software modules from high level requirements
2. Build prototype for a single software module
3. Implement any newly discovered requirements for module
4. Test software module
5. Repeats steps 2-4 until no more failures are found
6. Integrate module into core software build
7. Perform integration testing

Steps 2-7 are performed for each software module that is created in step 1.

For a single developer working alone it is unlikely that any form of software methodology would be used. The above list provides a systematic way for any developer working on their own, to not only follow a RAD approach but in the very least follow some predefined and proven approach. A detailed description of each step is outlined below.

1. Assign Software Modules from High Level Requirements

The principal requirement of this project was to build a quadrotor helicopter. From this principal requirement, the hardware was derived, and is described in Chapter IV .The software requirements are determined from both the electronic hardware (as the software provides an abstraction layer) and the principal requirement (the control system part of the software). Examples of software modules are:

- Serial port
- Proportional Integral Derivative (PID) control
- MotionPlus Sensor

2. Build Prototype for a Single Software Module

Once all the software modules are defined, each module can be individually built. A very simple list of requirements is derived for each module and implemented. This allows the design to be built gradually and the programming effort is focused on one area at a time.

3. Implement Any Newly Discovered Requirements for the Module

At the beginning of the development only a small subset of the full requirements are actually known. As a module is being constructed from basic requirements new requirements are discovered and implemented.

There are occasions when developing a module it is discovered another function is required to be implemented in an already existing module. As an example, if you are developing a debugging module that provides debug support it may be discovered a serial port function that can send a different data type is required. In this instance the new requirement is added to the serial port module.

4. Test Software Module

Once a module is completed, unit testing is performed. At this level of testing 100% code coverage should be achieved (i.e., all code is executed at least once). This testing is not emulated but instead actually performed on the target hardware with stimulated inputs and test-instrumented outputs. While this method requires more setup and implementation time than testing on the PC it provides a much higher level of confidence in the performance of the software.

5. Repeats Steps 2–4 Until No More Failures Are Found

Any faults found in the previous step are removed and testing is repeated until the module is working according to the requirements. This may require anything from small syntax corrections to complete rewrite of some functions, as well as the possible addition of helper functions.

6. Integrate Module Into Core Software Build

The complete and tested module is added to the core software build and recompiled together to ensure there are no linking errors. The main module is now modified to access the new modules functions.

7. Perform Integration Testing

Once the completed module has been integrated into the core software build, integration testing is performed to ensure there is no unexpected emergent behavior. As each module is added to the core software build and tested, the testing also ensures that other software modules have not regressed. One of the main criticisms of RAD is that of quality, because the completed software is ultimately based on a prototype [10]. This criticism is addressed by performing testing not only during the development of each prototype but also as each completed module is integrated into the main build.

B. SOFTWARE MODULES

The software modules assigned to the quadrotor are as follows:

- Main – The main entry point for the program
- CRC – Provides CRC16 data integrity functions
- mainLoop – The main control loop for the quadrotor
- mainLoopTimer – Controls timers for the loop trigger rate
- motionPlus – Hardware abstraction layer for the Wii MotionPlus
- PIDs – Implements the PID controllers for each axis of control
- PWM –Hardware abstraction layer for the PWM module
- Serial – Hardware abstraction layer for the serial port
- SPI – Hardware abstraction layer for the SPI port
- ipCapture – Hardware abstraction layer for the input capture module
- extFlash – Manages the external flash memory system
- engMode – Provide debug and tuning functions
- dataConversion – Provides data conversion helper functions
- lowpass_psv1 – FIR filter data structure for pitch axis
- lowpass_psv2 – FIR filter data structure for roll axis
- lowpass_psv3 – Fir filter data structure for yaw axis

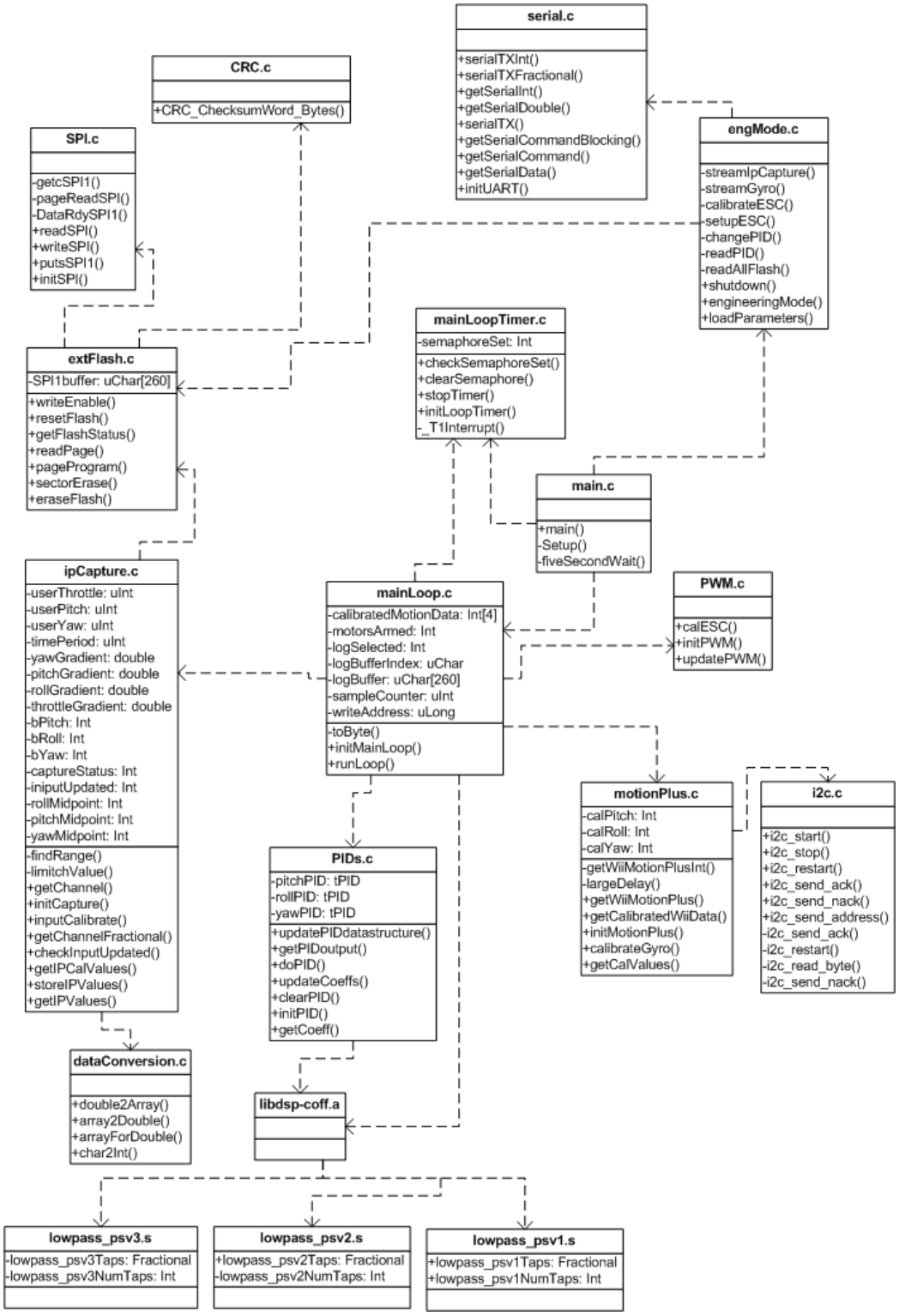

Figure 5. UML Diagram of quadrotor software

A UML class diagram is used in Figure 5 to show the function and file dependencies of the C code for the quadrotor.

C. FAULT DETECTION

The quadrotor relies on sensor data and stored calibration values to perform its function of providing augmented flight stability. If these sensors fail, or the calibration values are corrupted, the system needs to return itself to a safe state.

Calibration values for the transmitter, as well as PID values for the control loops, are stored in external flash memory. If these values are not read correctly by the DSP an undetermined state will result. For this reason, it is necessary to not only detect corrupted data but to additionally place the quadrotor in a safe state. A 16-bit CRC value is used to ensure data integrity when power is initially applied and the calibration values are read from external flash. If the CRC check determines that the data has been corrupted then the motors are turned off and placed in the disarmed state, the control loop is disabled, the user is notified via LED indication, and the software enters a continuous loop doing nothing. In this state, the quadrotor is unable to be flown.

Detecting failure of the MotionPlus module is not as simple. The difficulty in detecting a failure with the MotionPlus module is there is no data integrity checking data sent along with the gyroscope values, so it is not a simple case of performing a CRC check on the data. Through experimentation is was observed that if there was a failure of the MotionPlus, due to loss of power to the module or a severed communication line, the DSP would read a full scale value. At startup, the MotionPlus requires bias calibration (this is discussed more in the section on software development). If the module is working correctly the uncalibrated values from the MotionPlus will be somewhere around the middle of its possible range (~8192). At startup, the DSP checks the uncalibrated value from the MotionPlus, and if it is not approximately within the middle range then the same action for the erroneous CRC check explained above is initiated.

Without these safety-monitoring functions, it would be possible for the user to attempt to fly the quadrotor with no IMU information and erroneous PID values. This could result in catastrophic system failure.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SOFTWARE DESIGN

A. INTRODUCTION

The software for the quadrotor helicopter runs on “bare metal.” That is, the software runs directly on the dsPIC hardware with no dedicated operating system. All interaction with external devices required the construction of a specific software driver for that hardware module. Timing of critical functions is achieved through internal hardware timers that trigger an interrupt. This quadrotor implementation currently has 2300 lines of source code, not including DSP libraries.

B. SOFTWARE FLOW

1. Interrupts

The software uses interrupts to both control timing and perform management of some internal hardware peripherals.

The frequency at which the main control loop runs is controlled by an internal hardware timer. When this timer overflows, an interrupt is triggered. This interrupt sets a software semaphore that instructs the main software to execute the main control loop. The internal control loop is set to run at 200Hz. Ideally, the control rate should be much higher [8] but this is a limitation of the RC brushless ESC being used.

The other hardware interrupt is associated with the input capture module that reads the control information from the receiver. As there is more than one interrupt there are priorities assigned to each interrupt. While the interrupt associated with the input capture from the receiver is important, as this is the human input to the quadrotor, the control loop is more important. The control loop maintains the quadrotor stability and can perform this in the short term on its own without human intervention.

The signals captured from the RC receiver consist of a Pulse Position Modulated (PPM) signal for each channel received. The PPM signal is a pulse ranging from 1-2ms that is proportional to the stick position on the RC transmitter. A complete series of all channels is called a frame and is repeated approximately every 20ms. The last control

signal in the frame initiates an interrupt on both its rising and falling edges. Ideally, the interrupt would only occur on the falling edge of the last signal in the frame but this was not possible with the particular setup used and the input capture module of the dsPIC. When the falling edge interrupt occurs, the software reads the value captured by each channels register. This value is proportional to the pulse width of the incoming control signal. These values are scaled and converted into a fractional format so that the dsPIC hardware multiplier can handle them. The values are stored so they can be sampled by the control loop when required.

2. Flow Diagrams

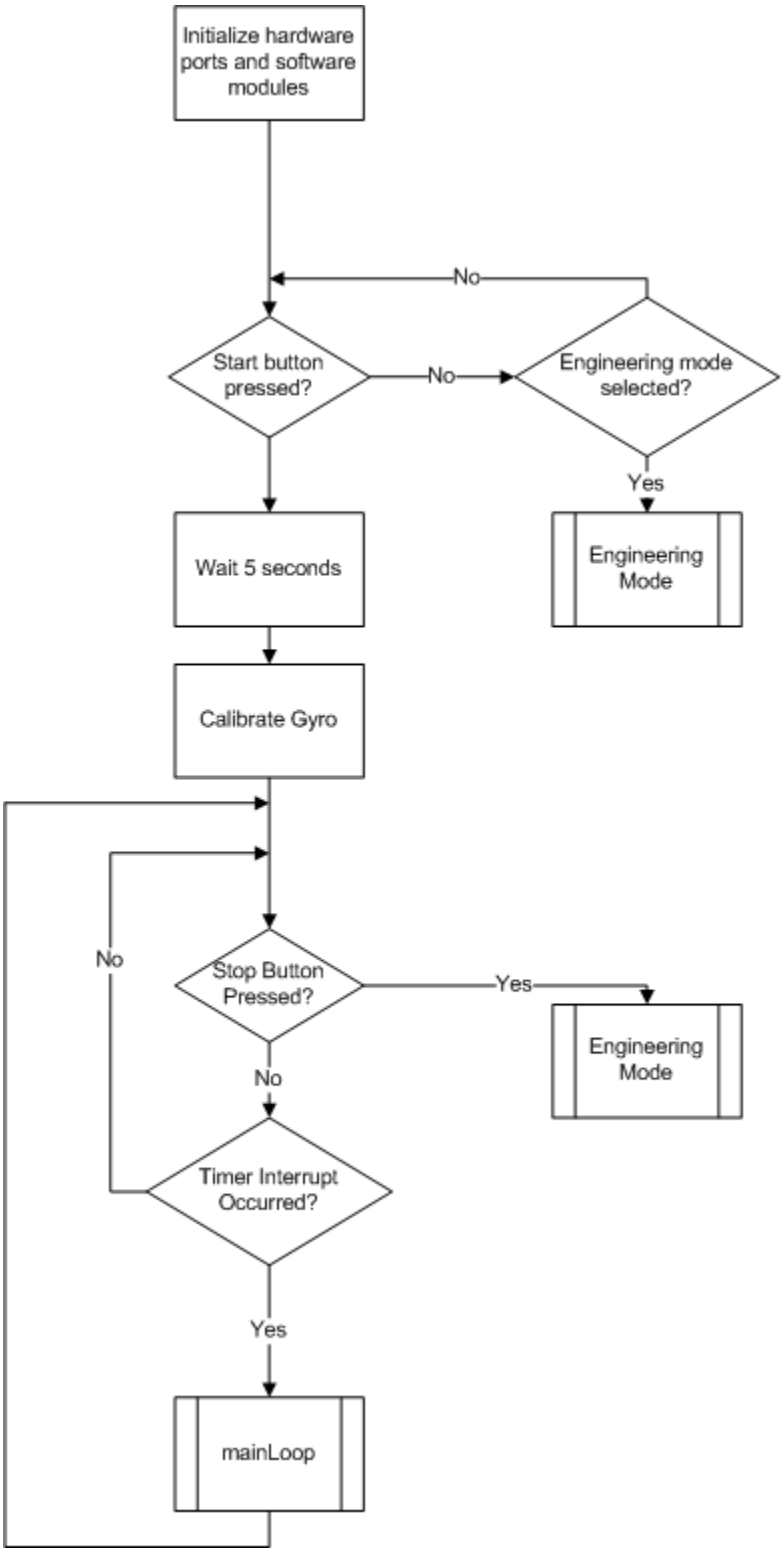

Figure 6. Software flow for main function

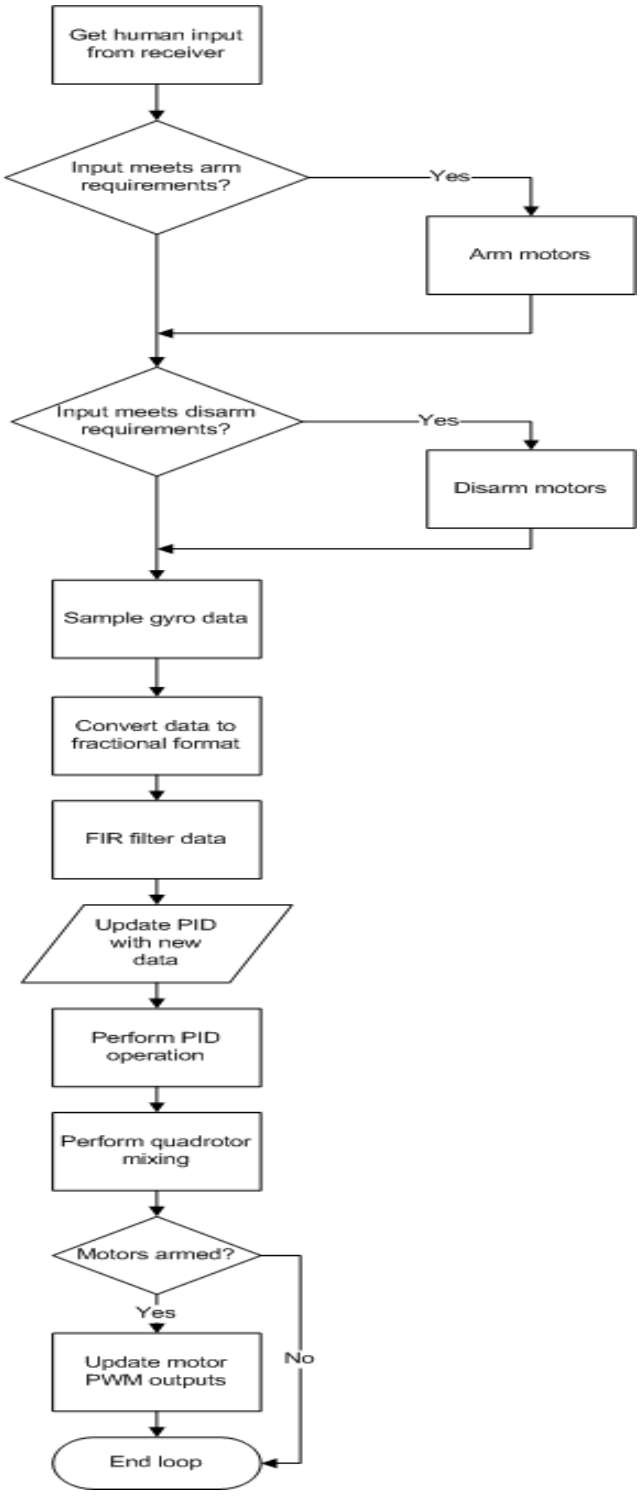

Figure 7. Software flow for mainLoop module

C. DIGITAL SIGNAL PROCESSING

For the on-board controller to provide augmented stability it must know how the aircraft is positioned in reference to the earth. This is termed aircraft attitude. For fully autonomous flight, the absolute attitude is required. While the end goal of this design is for the quadrotor to be fully autonomous, the initial requirement is to only provide augmented stability.

1. Augmented Stability Control Loops

For augmented stability, the quadrotor only requires information on its relative attitude. This greatly simplifies both the required sensors and the sensor processing. For relative attitude determination, gyro sensors can be used for each axis. The MEMS based gyro measures rotation rate. Using only gyro sensors there are two basic approaches to augmented flight control. The first is simply a rate limiter for each axis. This method measures the rotation rate from the gyro and a desired rotation rate from the controller and provides an error output that is the difference between these two signals. This error can then be applied to a PD or PID controller with the aim to minimize the rotation rate error. This control method has the effect of dampening the dynamic response of the aircraft so that it can be controlled by a human operator. This control method can be seen in Figure 8.

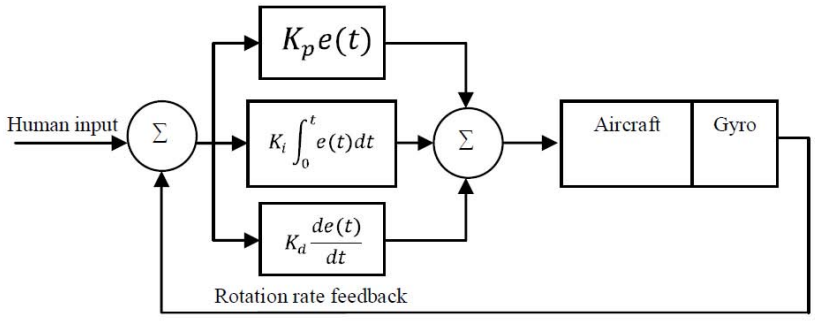

Figure 8. Rotation rate limiter for a single axis

The second approach that could be used for augmented flight control is a "heading hold" algorithm. The downside to the rotation rate limiter is that while it will reduce the dynamic response it does not hold the currently set aircraft attitude very well.

For the rotation rate limiter, if the controller encounters a disturbance, it will endeavor to minimize the rotation rate but the angular displacement from the initial attitude will move. This can be countered to some degree by using a full PID controller, as the Integral term will accumulate the error, but in practice, there is still significant deviation in attitude. The "heading hold" algorithm attempts to counter this problem by effectively not only dampening the rotation rate but also remembering the angular displacement encountered during a disturbance and then returning the aircraft to the initial attitude.

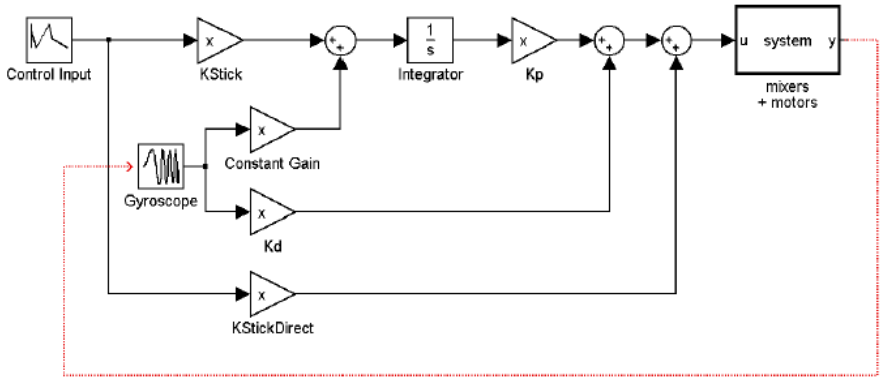

Figure 9. Heading Hold algorithm (From [8])

For most real-world control problems, the PID control loop provides a very robust and high performance controller. For this reason, the rotation rate limiter was selected for implementation on all axes. After in-flight evaluation of the performance of this implementation, the decision will be made on whether implementation of the heading hold algorithm is required.

2. Inertial Measurement Unit (IMU) Data Processing

Before the MEMS gyros can be used to provide the required information for a control system, the raw signal from the gyro must undergo some signal processing. MEMS gyros output a voltage that is proportional to the rotation rate they experience with a resting voltage approximately half way between the supply voltage and zero. The direction of the voltage deviation from the rest point indicates the direction of rotation.

For example, if the gyro resting voltage was 2.5V then a voltage of 3V may mean a clockwise rotation and a voltage of 2.0V would mean an anticlockwise rotation. This resting voltage is termed the gyro bias voltage and must be calculated each time the gyro is used. The difficulty in this arises because the gyro must be completely still during the calibration procedure and the voltage bias drifts with environmental conditions such as temperature. If an incorrect bias is used by the control system then the system will think that there is a constant rotation rate when there is no actual rotation at all.

The Wii MotionPlus uses two MEMS gyro parts to obtain 3 degrees of freedom [11]. An InvenSense IDG-600 [12] is used to provide both pitch and roll gyro information, all in a single package. A single axis gyro from Epson Toyocom [13] provides yaw information. While the MotionPlus performs analog to digital conversion for each gyro axis, it does not calculate the bias point and this has to be performed by the quadrotor control system. The bias point is obtained by averaging a large number of gyro samples while the quadrotor is left stationary.

To assess the amount of bias drift experienced by the MotionPlus device, gyro samples for each axis were logged at 200Hz over a period of 5 minutes while the quadrotor remained completely still. The gyro data is a 14-bit value and hence has a calibrated range of +/-8192. Figure 10 shows the samples collected from the pitch gyro with a fitted line that shows the bias drift. In this figure, the bias has already been calculated and applied, hence, the data is initially centered around zero.

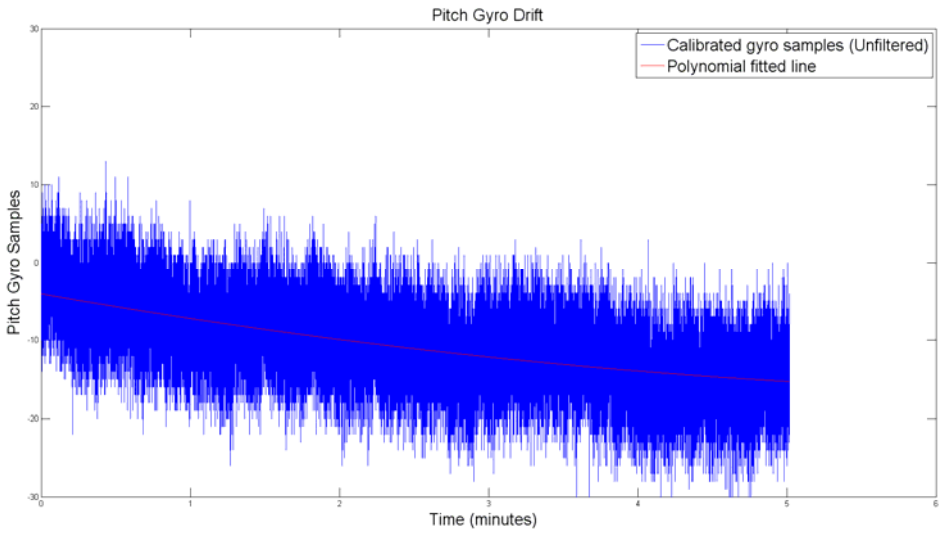

Figure 10. Pitch gyro bias drift

There is considerable noise present on the gyro signal along with an obvious bias drift. The bias drift may seem small compared to the maximum possible signal of the gyro (+/-8192) but if this signal is integrated to calculate angular displacement in degrees (as is the case when the Heading Hold or full PID algorithm is implemented) then the error continually grows. Figure 11 shows the angular drift over time for the pitch gyro.

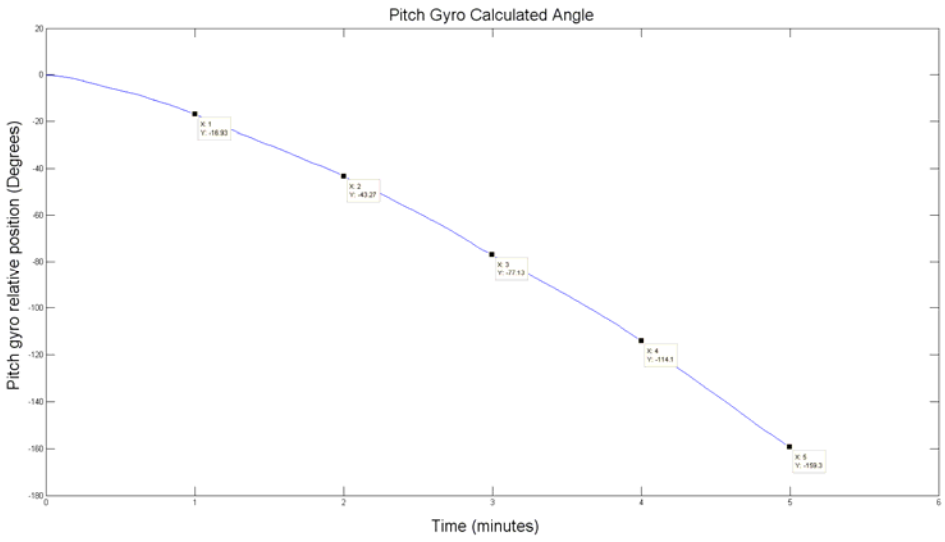

Figure 11. Pitch gyro angular drift

From Figure 11, the pitch gyro can be seen to have drifted -159.3 degrees in 5 minutes. This is a drift rate of 0.53 degrees/second. Due to such a high drift rate, it will be impossible for the control system to maintain a particular attitude for even a brief period while just using MEMS gyros. Despite this drift, the information from the MEMS gyro will be enough to provide approximate relative attitude to the controller, so that it can dampen the dynamic response.

The yaw gyro bias drift and angular drift can be seen in Figures 12 and 13, respectively. Figure 12 clearly shows a much lower bias drift than the pitch axis gyro, which will contribute to a much better angular drift figure, as seen in Figure 13. From Figure 13, the gyro drift for the Yaw axis can be seen to have moved 5.322 degrees in 5 minutes, a drift rate of 0.018 degrees/second. The lower drift rate of the yaw gyro is mostly attributed to fact that the yaw gyro is rated at 100 deg/sec while the pitch and roll gyro is a 2000 deg/sec gyro.

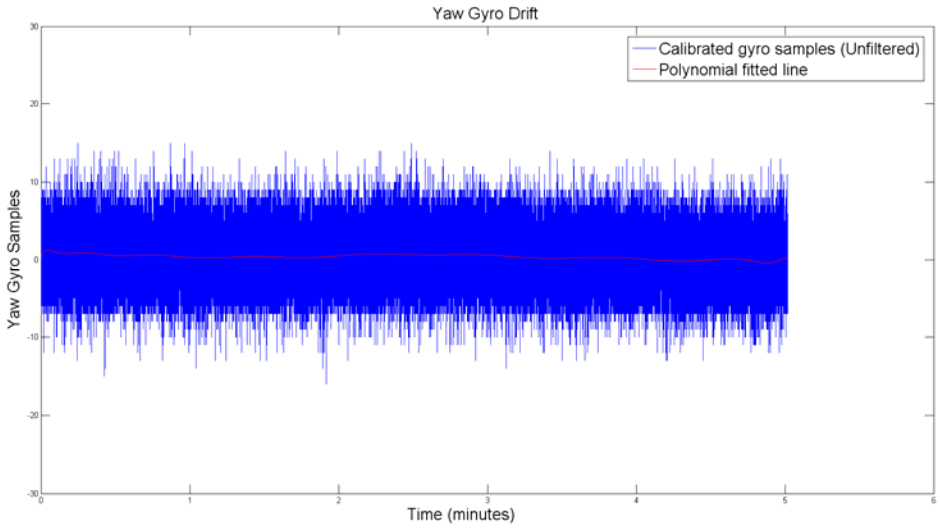

Figure 12. Yaw gyro bias drift

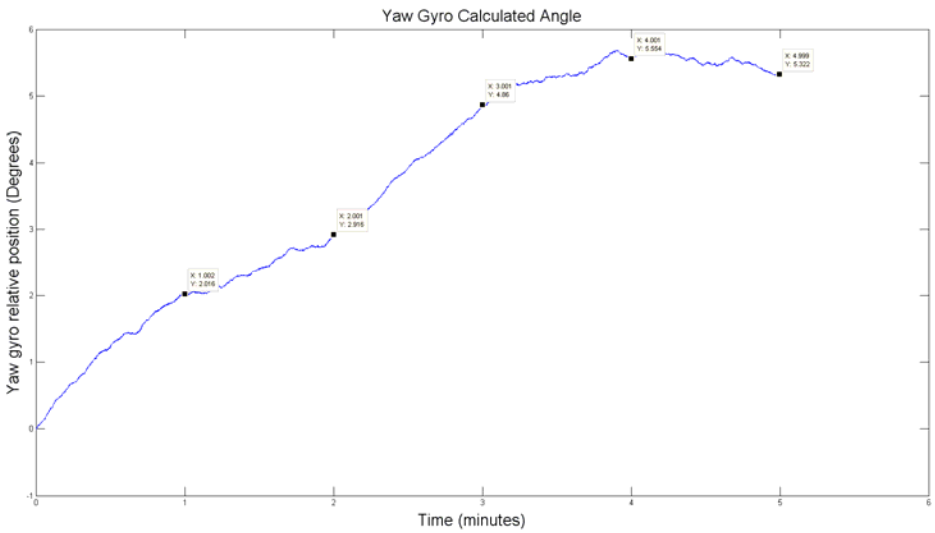

Figure 13. Yaw gyro angular drift

The predominant cause of drift in a MEMS gyro is temperature change [9]. For this reason, some implementations use a pre-computed look up table to provide continual updates to the bias level depending on the current temperature. Due to manufacturing differences between devices of the same part number, it is not practical for manufacturers to provide accurate temperature compensation information for each serial number. To obtain accurate temperature compensation data, each gyro is required to have its bias voltage measured and tracked while within a temperature-controlled environment. Currently there is no temperature compensation performed on the gyros used for this quadrotor. This could be an area of further research if time constraints allow.

3. Finite Infinite Response (FIR) Filters

The noise present in the gyro signal while the quadrotor is operating has both electrical and mechanical components. This noise should be attenuated as much as possible to provide the controller with a high Signal to Noise Ratio (SNR) and, hence, the most accurate estimate of rotation rate from the gyros.

The MotionPlus gyro module provides data in 14-bit format with a voltage reference of 3.3V. The voltage resolution is calculated below. Given such a low voltage resolution, there will be considerable electrical noise.

$$2^{14} = 16384$$
$$\frac{3.3}{16384} = 201.41\mu V$$

The mechanical component of noise comes from the four DC motors. This will be the largest component of noise and must be filtered. FIR filters were used for this purpose.

A FIR filter is a type of digital filter that uses the mathematical process of convolution to remove unwanted parts of a signal. A block diagram of the basic FIR structure is shown in Figure 14.

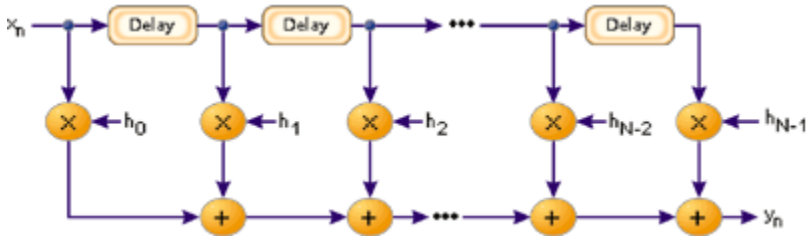

Figure 14. FIR filter basic structure (From [15])

The constant values of h_k are termed the filter coefficients and are selected to obtain the desired filter response. Filter coefficients are often automatically generated from a digital filter design program once filter parameters are selected.

In order to design the FIR filter the desired cutoff frequency is required. The cutoff frequency must be carefully selected so that noise is filtered out while the dynamic motion of the aircraft is not. To obtain the frequency of the noise present in the gyro signal, the gyro data was sampled while the quadrotor was fixed to a test rig that only allows a single degree of freedom movement and the motor speed was increased to take off power. Figure 15 shows the spectral view of the sampled pitch gyro data during the aforementioned test.

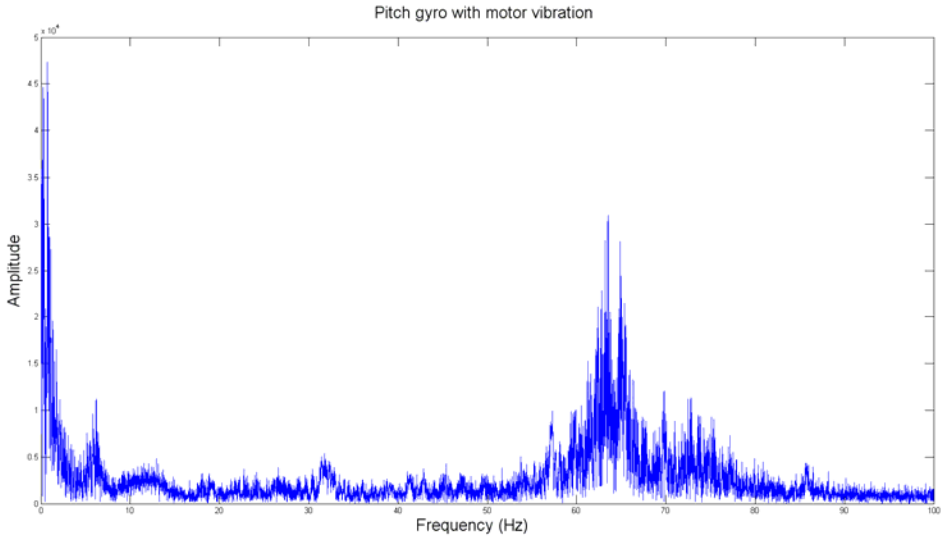

Figure 15. Pitch gyro spectrum with induced mechanical vibration

Figure 15 is obtained by performing a Fast Fourier Transform (FFT) of the pitch gyro samples. Most of the mechanical noise can be seen between ~50-80Hz. The lower frequency components (<10Hz) will contain the wanted gyro rate information. For this reason a cutoff frequency of 8-10Hz was selected.

To generate the FIR filter coefficients the Microchip dsPIC Lite program was used. Figure 16 is a screenshot of the filter design program showing the designed filter. This is a 16-tap FIR filter that will attenuate the noise that exists at ~40-80Hz by at least -50dB. During operation, the dsPIC controller will filter all three gyro axis at each control iteration. This is a considerable computational load for a low cost embedded system but the dsPIC architecture is designed for a highly efficient implementation of FIR filtering due to the hardware multiply accumulate engine and DSP architecture.

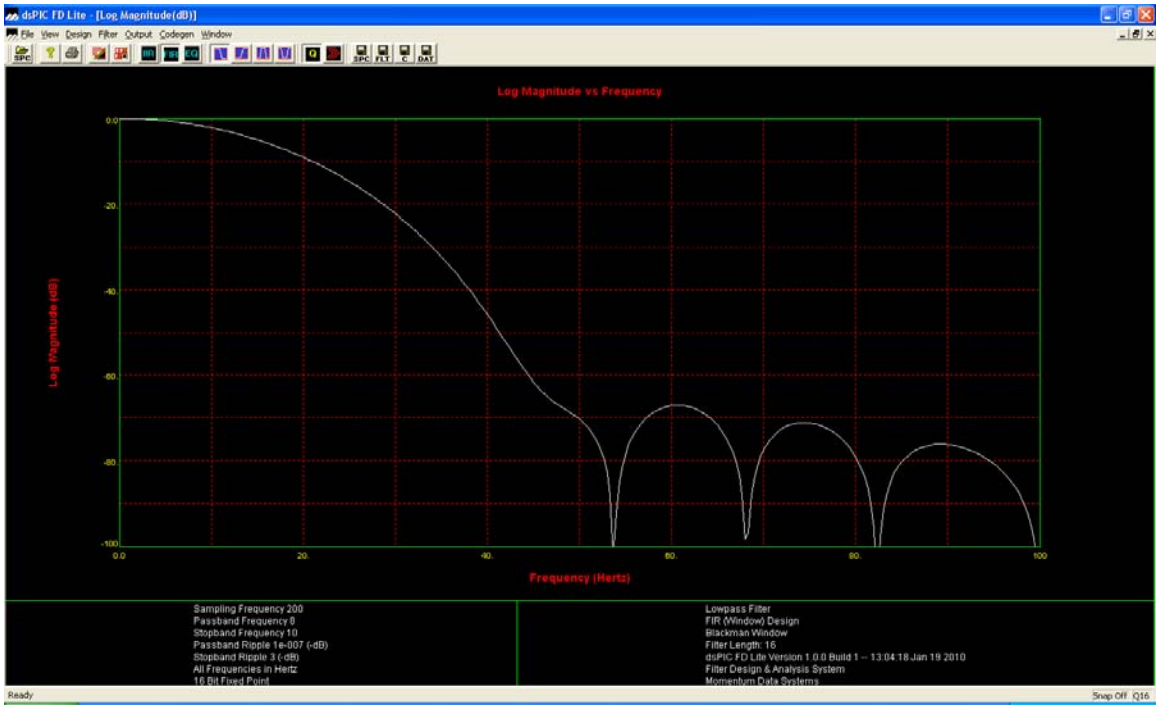

Figure 16. dsPIC Lite filter program

To test the filter implementation the same vibration test as mentioned above was performed again only this time the FIR filter was operating on the dsPIC controller. Both the raw input samples to the filter as well as the filtered output samples were logged. Figure 17 shows the unfiltered roll gyro samples during the test while the quadrotor moves back and forth in the roll axis. The noise can be seen superimposed over the rate information from the gyro. Figure 18 shows the filtered gyro samples from the FIR filter. It can be seen from the filtered gyro samples that the filter is working well and removing most if not all of the mechanical noise.

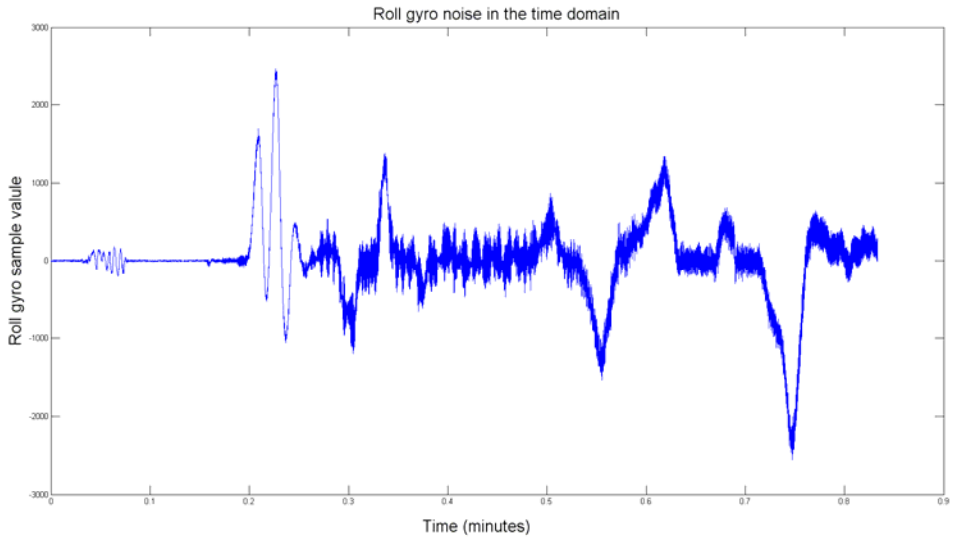

Figure 17. Roll gyro samples showing mechanical noise

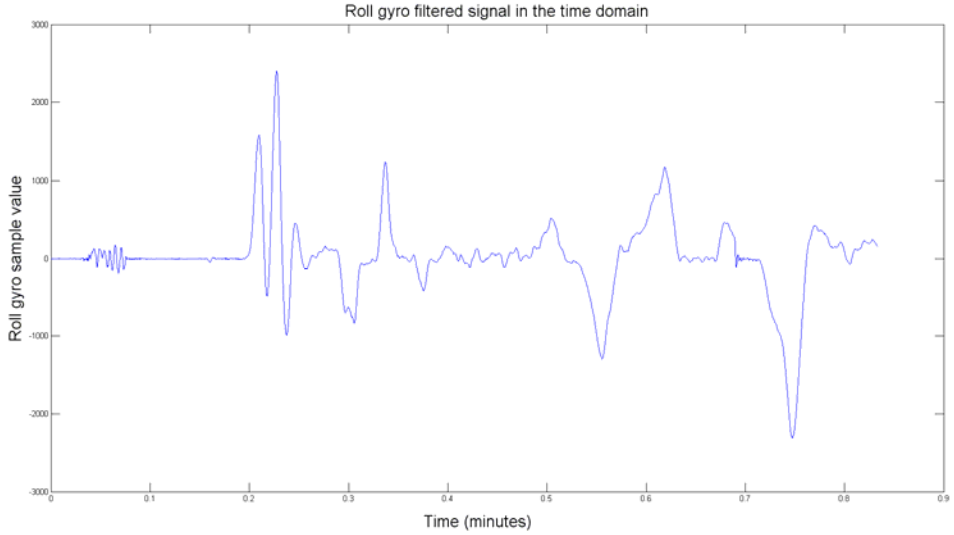

Figure 18. Filtered roll gyro samples

D. SOFTWARE TESTING

Testing was performed at each stage of the design to ensure correct operation of each module and to confirm that the software did not regress when further changes were made. Once all software was complete, integration testing was performed before adding the motors and speed controllers to the existing hardware. This was to ensure that the quadrotor responded as expected and, in particular, that no un-commanded motor operation was encountered. This was a specific safety concern as a moving propeller has the potential to cause injury.

Given the number of sensors and control inputs to this controller, the input space for testing is very large (i.e., if you were to write a list of all the possible combinations of inputs to the controller the list would be very large). This makes it impossible to test the entire input space so a more functional approach is taken. Testing was performed by instrumenting the code [16]. By using this method, data of interest is sent out the serial port in real time to be displayed by a program on the PC. The instrumenting of code provided the following data over the serial port:

- Speed command being sent to all motor controllers
- Unfiltered gyro data
- Filtered gyro data

A LabVIEW [17] program was created to provide a graphical representation of the instrumented data in real time. A screenshot from the LabVIEW test program can be seen in Figure 19.

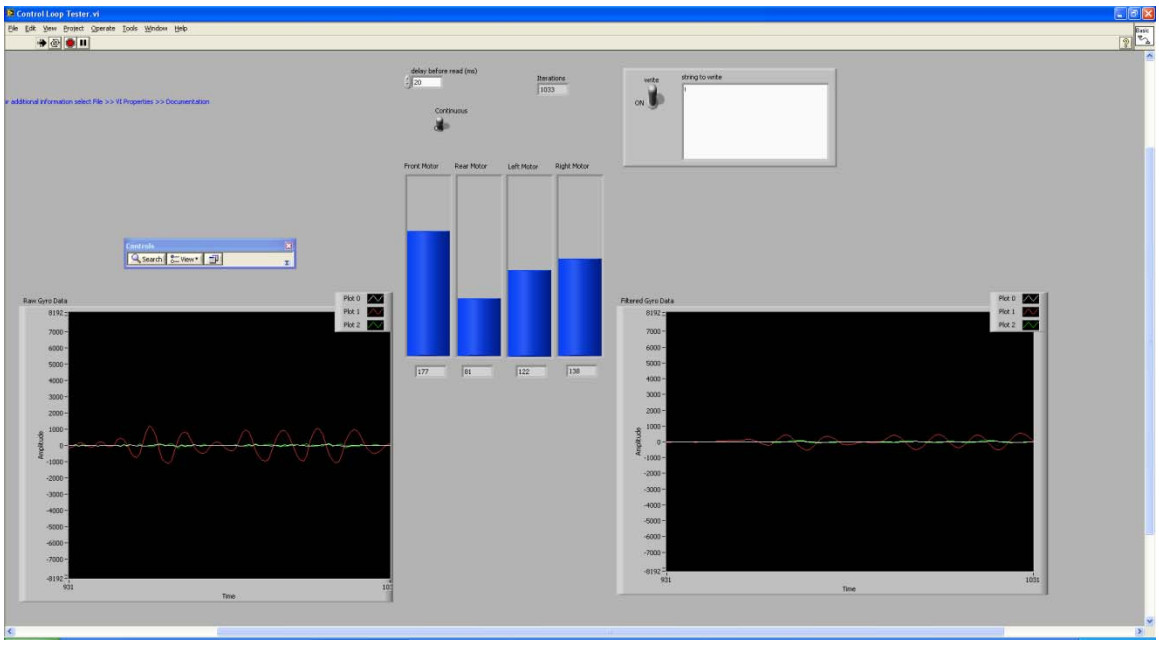

Figure 19. LabVIEW test program screen capture

For testing with LabVIEW, the internal timer interrupt in the on-board controller was disabled so that the control loop can be triggered from LabVIEW. The control loop rate can be changed by adjusting the "delay before read" data box in the LabVIEW Graphical User Interface (GUI). A serial representation of the PWM command sent to each motor controller can be seen in the middle of the screen shot of Figure 19. For each iteration of the control loop, triggered by LabVIEW, the on-board controller performs the functions of the main control loop and sends an update of each instrumented parameter back to the PC for display.

Using this display, a complete integration test of all software can be performed. Several tests were performed to confirm correct operation of each software module. Once all tests are complete, a high level of confidence in the function of the software is achieved. Specific tests are discussed below.

1. Test 1

The first test to be performed was simply to have the aircraft stationary while moving the RC transmitter sticks in each of their own axis. While performing this test, the motor outputs on the LabVIEW program are observed to confirm correct behavior.

The bar graph representation on LabVIEW should move smoothly with control input from the RC transmitter sticks and the correct quadrotor motor mixing should also be confirmed. This test confirms correct operation of the following software modules:

- ipCapture
- serial
- mainLoop

2. Test 2

In this test, the throttle on the RC transmitter is set to half way and the aircraft is rotated back and forth individually around its 3 axis. The behavior observed on the LabVIEW program should indicate that the controller is sensing the movement and is trying to compensate for the induced attitude change. For example, if the aircraft is rolled to the right the right motor speed will be seen to increase while the left motor speed will be seen to decrease. This will have the effect of the aircraft acting to oppose the movement. This is carried out for each direction of rotation around each axis. This test confirms correct operation of the following software modules:

- ipCapture
- serial
- mainLoop
- motionPlus
- PID (To a limited degree)

3. Test 3

In this test, the aircraft is kept stationary on the ground and the PWM outputs from the dsPIC are connected to an Oscilloscope. The PWM outputs are observed from when the quadrotor is started until the motors are armed to ensure that the PWM is commanding a motor off condition irrespective of the setting of the RC transmitter sticks. Once the motors have been armed the PWM outputs are observed to confirm that they are proportional to the output that is indicated in the LabVIEW display. The motors are then placed again in the unarmed state, the throttle on the controller is increased and the motor

PWM outputs are observed on the oscilloscope to ensure they are maintaining the motors in the off state. This confirms correct operation of the following software modules:

- PWM
- engMode
- ipCapture

4. Test 4

The purpose of this final test was to confirm the correct timing of the operations within the dsPIC. The main control loop executes at a rate of 200Hz, this is a period of 5ms. The dsPIC has <5ms to perform all operations within the control loop before it is required to perform these operations again with new data. If the dsPIC is unable to perform all operations within the 5ms timeframe, then the system will become unstable and most likely crash.

The most accurate way of measuring and confirming the correct timing was to place commands in the code that toggle one of the output pins when the main control loop starts and stops. Using an oscilloscope, both the main control loop period and execution time can be monitored.

Results from the test concluded that the main control loop was executing once every 5ms as required. The oscilloscope also showed that it was taking 1ms to complete the main loop when not writing logged data to the flash memory and 2ms to complete when writing a page of flash data. This gives a maximum CPU usage of 40%. The extra 1ms required for writing to the flash device could mostly be eliminated by using the Direct Memory Access (DMA) hardware within the dsPIC to perform the data transfer but this was not implemented due to time constraints. Using the DMA hardware to perform external flash memory data transfers would reduce the CPU load to 20%. At least half of this 20% CPU usage is devoted to waiting for the MotionPlus device to provide another sample. This could further be reduced by implementing a Real Time Operating System (RTOS) to make use of the CPU while it waits for the MotionPlus.

E. PC BASED CONFIGURATION SOFTWARE

For the configuration of internal settings and control parameters, as well as providing assistance in debugging, a PC based configuration program is highly desirable. A PC based GUI was designed to provide the following functionality:

- Fast adjustment of internal PID values
- Initiate and confirm RC transmitter calibration
- Initiate and confirm gyro calibration
- Calibrate ESCs
- Manage external flash memory data

A screen shot of the PC GUI can be seen in Figure 20.

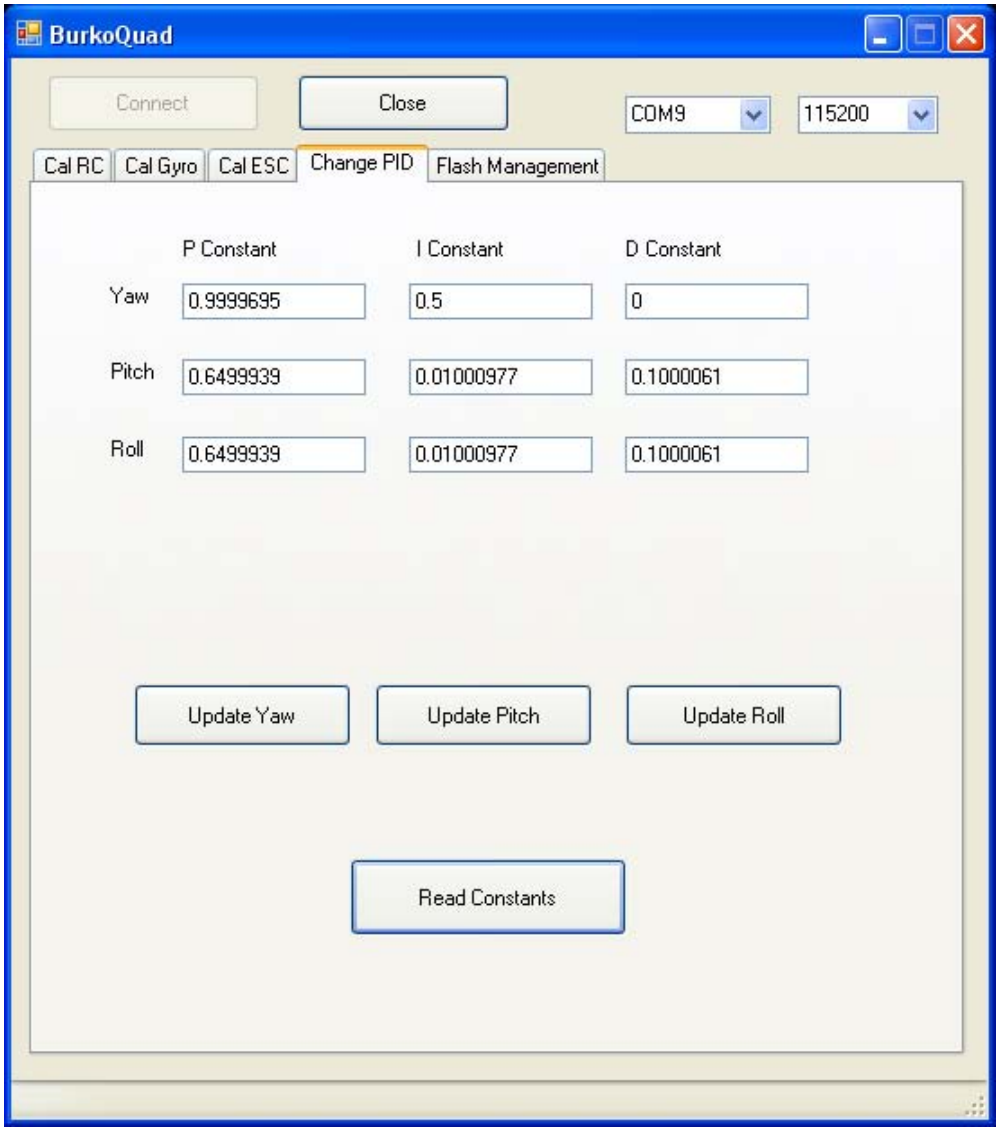

Figure 20. PC configuration GUI

The GUI was built in Visual C++ to enable rapid development. The functions performed by the GUI are described blow.

1. Fast Adjustment of Internal PID Values

During the PID tuning process, it is anticipated that many changes to PID values will be required. It is not realistic to make these changes in firmware and then recompile the code each time a change is made. Therefore, this function provides a means of making “on-the-fly” changes to PID values. As a means of checking that changes have

been made correctly and sent without error to the dsPIC, the PC program can also read the current PID values. To make changes to PID values, the motors on the quadrotor must be in the disarm state. The serial cable is then connected to the quadrotor and the changes can be made quickly.

2. Initiate and Confirm RC Transmitter Calibration

This function initiates the RC transmitter calibration process. This is required so the full range of each control stick can be used and that the resting points of the joysticks are read as zero by the on-board controller. After a calibration is performed, the calibration constants are sent to the PC GUI. This can be used to ensure correct calibration and for debugging purposes.

3. Initiate and Confirm Gyro Calibration

While the on-board controller performs gyro calibration as a part of its startup routine, there can be considerable drift in the gyro bias due to temperature. For this reason, it is beneficial to have a manual ability to initiate a gyro calibration routine. As this function also provides a means of confirming correct operation of the MotionPlus device, this was another reason for its use in development. After completing a gyro calibration, the dsPIC will send the new bias values back to the PC for display.

4. Calibrate ESCs

When new ESCs are added to the quadrotor, they require their PWM input range to be calibrated so control over their entire power range is allowed. This also ensures that the same power is applied to each motor for a given PWM value. This function allows the on-board controller to directly generate the required PWM sequence for ESC calibration.

5. Manage External Flash Memory

The external flash memory is used to store configuration and control parameters as well as being used for data logging. This function reads the contents of the memory and stores it in a file on the PC. Log data can then be imported to another program for analysis. There is also a single button to erase flash data.

Once transmitter calibration and PID values have been updated, there is also an option to write these values to the parameter configuration section of the external memory. This saves the user from having to calibrate the transmitter and enter PID values each time the quadrotor is used.

VII. TUNING AND TETHERED TESTING

A. INTRODUCTION

To ensure that the quadrotor is stable enough to control before the first flight, the PID control loops require tuning. Some methods available for tuning PID controllers are the Ziegler-Nichols method, software based, or manual tuning. Manual tuning was selected for the initial tuning method for this project, as it is easy to set up and provides visual confirmation of the correct implementation of the PID controller.

B. TEST RIG

For PID tuning, each axis was individually isolated and separately tuned. The test rig used for pitch and roll can be seen in Figure 21. The assumption when using this rig to perform testing is that it provides minimal dampening of the dynamics and so provides a reasonable approximation to dynamic response of an axis in flight.

Figure 21. Quadrotor test rig

The process for manual PID tuning starts by gradually increasing the proportional gain until the system experiences some minor oscillations before settling. Next, the integral gain is increased to remove the steady state error. At this stage, the quadrotor will respond rapidly to an input from the RC transmitter and will move to the new commanded position. The integral term now present in the PID loop means that the quadrotor holds the currently set attitude more accurately, as any drift from this point that is detected by the gyro is integrated so that it "remembers" the angle it has drifted. Finally, the differential term is increased to remove the slight overshoot and oscillation.

During tuning, the behavior described above could be observed for each parameter of the PID controller. This provided confidence that the PID implementation was correct and also gave an insightful understanding of the workings of PID controllers.

For tuning the yaw axis, a "lazy susan" was used. This allowed the quadrotor to move around its yaw axis while fixing both the pitch and roll axis. The same method for tuning the pitch and roll was also used for yaw.

This manual tuning is only intended as an initial method to get a rough estimate of the required PID values. Once the quadrotor has some accumulated flight time, the data logging function will be used to log both input to motor controllers as well as gyro outputs. If time permits, this data will be used for system identification to develop a black box model of the quadrotor. From this model, more precise control parameters could be selected.

C. TETHERED TESTING

If the quadrotor were to crash during first flight, this could lead to catastrophic damage of the on-board controller and the quadrotor frame. This would considerably delay the project. For this reason, all practical steps were taken to ensure that the control system was performing satisfactorily before the first flight. To ensure that the control system was stable enough for flight, while allowing for an unstable system, the quadrotor was tethered from the ceiling and placed on a small takeoff platform. Using this tether, the quadrotor can be flown in a very limited area. If a failure occurs, the tether stops the quadrotor from striking the ground.

The tether is mounted on a pulley system with a counterweight provided, which allows just enough weight to remove all slack from the tether. This prevents the tether from becoming tangled in the propellers of the quadrotor. Using the tethered setup, the quadrotor was successfully hovered in a small area. This provided more confidence that the PID values discovered in the tuning phase are sufficient and flight tests can now be carried out.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. FLIGHT TEST

A. INTRODUCTION

The first flight test was carried out on a clear day with light wind. The quadrotor flew without any problems. The stability requirement was met as the quadrotor could be controlled by a human operator. Figure 22 shows logged data for the human input control for the first flight. An explanation of the logged data is as follows.

The first green spike from the user places the motors in the armed state. The throttle (yellow) is then gradually increased till the aircraft is in a hover position. For the first flight, the pitch and roll controls were quite sensitive and required some familiarization. This can be seen by the pitch (red) control during the first part of the flight where overcompensation occurred on a few occasions. The two negative yaw (green) control inputs are when the quadrotor was aligned so that the tail points directly toward them. From approximately 0.8 to 1 minute, there are very small adjustments to pitch and roll to maintain the quadrotor hovering over the same position. From 1 minute onwards, the quadrotor was allowed to drift from the current position while correction was only provided if there was any drift from a flat pitch and roll.

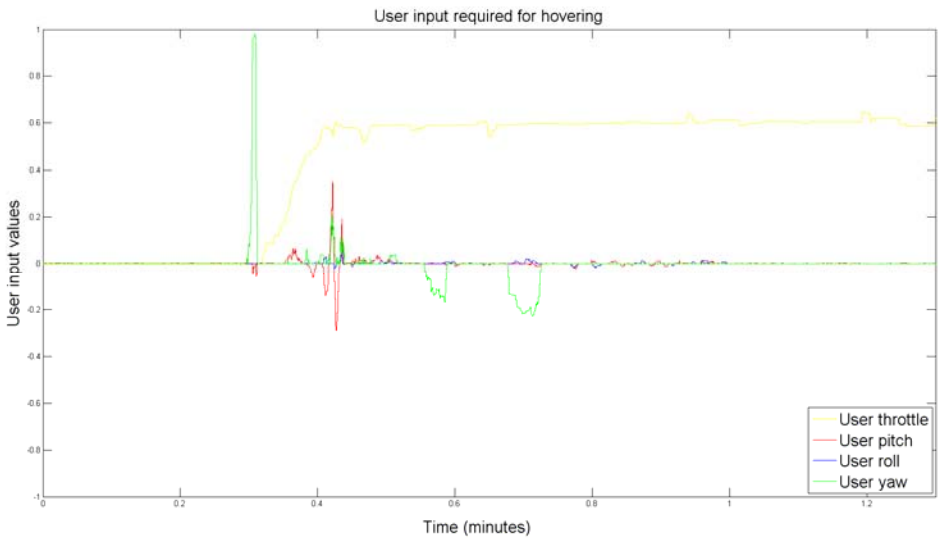

Figure 22. User input for first flight

As can be seen from the figure, there is very little, if any, input required to maintain a constant attitude. Given that GPS is required to maintain a position when using MEMS IMUs (due to the excessive drift), and only MEMS gyros are used in this implementation, this platform is very stable. During light gusts of wind, the quadrotor would noticeably compensate for the perturbation. When use of the control sticks of the transmitter stops, the quadrotor does not auto-level at this stage. This is because the gyros only provide relative position information and the controller is not aware of the current absolute attitude. With the addition of accelerometers, the quadrotor will provide an auto-leveling feature and will provide an enhanced level of stability.

Flight endurance is currently estimated at 8 minutes for a 2,000mAh battery, compared to the Ascending Technologies quoted flight time of 20 minutes on a 2,100mAh battery for their research pilot aircraft. The difference in flight times is mostly attributed to weight, as the Ascending Technologies research pilot weighs 500g including the flight battery, while the quadrotor designed in this thesis currently weighs 1236g including the flight battery. The frame of this design is the largest contributor to the total weight of the aircraft, as it is constructed of aluminum, PVC, and polycarbonate. The frame was selected for its simple construction and was only intended as an interim measure. A lighter frame design would increase flight time and allow larger loads to be carried.

Figure 23. Quadrotor in flight

THIS PAGE INTENTIONALLY LEFT BLANK

IX. CONCLUSION

The goal of this thesis was to produce a low-cost quadrotor research platform by maximizing the use of COTS equipment. The total cost of the current platform is $313.60 (see Table 2). This was achieved though the set objective of using mass-produced COTS parts such as the Wii MotionPlus, low-cost brushless motors and controllers. While in its current form, the quadrotor is not fully autonomous. The only additional hardware required for autonomous flight is a GPS, 3-axis accelerometer, and 3-axis magnetometer. These items could be purchased for approximately $150. These additional components would add 30g to the total weight of the quadrotor and have a negligible affect on flight duration.

Flight test data from Chapter VIII provides evidence of a stable platform. This is important when being used for research, as it will limit the time spent learning to fly the quadrotor.

While the quadrotor designed in this thesis consists of many parts, both hardware and software, it inherits most of its ability through software. Not surprisingly, the software development side of this thesis consumed 95% of the effort. A RAD methodology was adopted for software development, which allowed a fast start to software development and worked well to discover missing requirements as the development progressed. This is evidenced through the fact that, before starting the project, the only initial requirement was “to build a quadrotor helicopter” and, at the completion of the project, the quadrotor was flying under human control. One of the main strengths of the RAD methodology applied in this design was the continual testing performed at each iteration in the development of a software prototype. Once all the software prototypes were combined into a single build, there was no unexpected emergent behavior. It is highly unlikely this would be the case when using a traditional Waterfall model.

Products that have a relatively small software requirement can be developed by a single developer. In this instance, it is unlikely that a structured software methodology

would be used. The RAD methodology presented in this thesis is ideal for a single developer to follow as it is simple and encourages a short planning stage. The final product of this thesis is a testament to its success. A modular software design supported the RAD methodology and also provides a more understandable and modification-friendly code. Having the software source code, and hardware schematics from this design, allows greater flexibility in further research topics with this platform.

The first additional improvement that could be made to this design is a lighter frame. While the current 8 minutes' duration on a 2,000mAh battery is still useful, the lighter frame would add considerable endurance to the platform. The next step is to achieve full autonomy. This would require the addition of GPS, magnetometer, and 3-axis accelerometer along with accompanying software. The final step would be to add a powerful embedded processor such as the beagleboard from www.beagleboard.org. This board would communicate with the dsPIC over a simple serial interface and would provide enormous scope for additional research. Having the dsPIC handle the real time processing of maintaining position and attitude allows the higher-level processor to perform other supervisory functions such as image processing.

While fully autonomous flight has not been achieved, this goal is not too distant. Development of this quadrotor will continue after completion of this thesis to achieve fully autonomous flight and waypoint navigation.

LIST OF REFERENCES

[1] Wikipedia [Online]. http://en.wikipedia.org/wiki/Miniature_UAV (2010, January).

[2] Wikipedia [Online]. http://en.wikipedia.org/wiki/Quadrotor (2009, October).

[3] S. Bouabdallah et al., "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor," Lausanne, Switzerland.
.

[4] Samir Bouabdallah, Murrieri Pierpaolo, and Siegwart Roland, "Design and Control of an Indoor Micro Quadrotor," Autonomous Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland.

[5] Elaine M Hall, *Managing Risk*. Kansas: Addison Wesley, 2007.

[6] Department of the Prime Minister and Cabinet. Department of the Prime Minister and Cabinet. [Online]. http://www.dpmc.gov.au/implementation/policy.cfm (2009, February).

[7] Wikipedia. [Online]. http://en.wikipedia.org/wiki/MEMS (2010, Janurary).

[8] Microchip Technonlogy. (2006) dsPICDEM 80-Pin Starter Development Board User's Guide. pdf.

[9] Aeroquad. [Online] http://aeroquad.info/bin/view/Main/BuildInstructions_v11 (2009, June).

[10] Favonian. Wikipedia. [Online]. http://en.wikipedia.org/wiki/Rapid_application_development (2010, February).

[11] Roger A Muller, "Extreme Programming in a University Project," *Extreme programming and agile processes in software engineering*, vol. 5, pp. 312–315, 2004.

[12] Gurdan Daniel et al., "Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz," in *2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, 2007, pp. 361–366.

[13] Nintendo. Nintendo.com. [Online]. http://www.nintendo.com/wii/what/accessories/wiimotionplus (2010, January).

[14] Invensense. Invensense. [Online]. http://invensense.com/mems/gaming.html (2010, January).

[15] Epson Toyocom. Epson Toyocom. [Online]. http://www.epsontoyocom.co.jp/english/product/Sensor/set01/xv3500cb/index.html (2010, January).

[16] Kirill Shcheglov, Christopher Evans, Roman Gutierrez, and Tony K. Tang, "Temperature dependent Characteristics of the JPL Silicon MEMS Gyroscope," Jet Propulsion Labratory, California Institute of Technology, Pasadena.

[17] Brian Wagner and Michael Barr, "Introduction to Digital Filters," *Embedded Systems Programming*, pp. 47–48, December 2002. [Online] http://www.netrino.com/Embedded-Systems/How-To/Digital-Filters-FIR-IIR (2010, January).

[18] Paul Ammann and Jeff Offutt, *Introduction To Software Testing*. New York, United States: Cambridge, 2008.

[19] National Instruments. National Instruments. [Online] http://www.ni.com/labview/ (2010, January).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California